UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

CURSO DE GRADUAÇÃO DE ENGENHARIA ELÉTRICA

MATHEUS VINÍCIUS MORAIS LEITE

AUTOMAÇÃO DE UMA MÁQUINA ENVASADORA: ESTUDO, SIMULAÇÃO E DESENVOLVIMENTO DE UMA INTERFACE HOMEM-MÁQUINA

FORTALEZA

2014

MATHEUS VINÍCIUS MORAIS LEITE

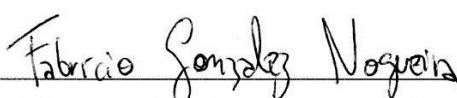
AUTOMAÇÃO DE UMA MÁQUINA ENVASADORA: ESTUDO, SIMULAÇÃO E DESENVOLVIMENTO DE UMA INTERFACE HOMEM-MÁQUINA

Monografia submetida ao Curso de Engenharia Elétrica da Universidade Federal do Ceará como parte dos requisitos para obtenção do grau de Graduado em Engenharia Elétrica.

Área de concentração: Automação e controle.

Orientador: Prof. Dr. Fabrício Gonzalez Nogueira.

FORTALEZA

2014

MATHEUS VINÍCIUS MORAIS LEITE

AUTOMAÇÃO DE UMA MÁQUINA ENVASADORA: ESTUDO, SIMULAÇÃO E DESENVOLVIMENTO DE UMA INTERFACE HOMEM-MÁQUINA

Esta monografia foi julgada adequada para a obtenção do grau de Graduado em Engenharia Elétrica e aprovada em sua forma final pelo Orientador e pela Banca Examinadora.

Aprovada em: 13/06/2014.

BANCA EXAMINADORA

Fabrício Gonzalez Nogueira

Prof. Dr. Fabrício Gonzalez Nogueira (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Paulo Peixoto Praça
Universidade Federal do Ceará (UFC)

Aloísio F. Dias

Eng. Aloísio Fernandes Dias
Universidade Federal do Ceará (UFC)

Aos meus pais, Marcos e Telma.

À minha família.

Aos meus colegas de curso.

RESUMO

Este trabalho apresenta o estudo e simulação de um sistema de automação industrial para um processo de envase de margarina de uma indústria alimentícia do estado do Ceará. O sistema estudado tem como principal componente um controlador lógico programável (CLP) que executa um programa desenvolvido em linguagem *ladder*. A lógica de automação embutida no CLP foi simulada por meio de um conjunto de ferramentas computacionais, o que permitiu o estudo e testes sem a necessidade de conexão com o sistema de automação real. Além disso, para fins de operação do sistema simulado, neste trabalho também foi desenvolvida uma interface homem-máquina (IHM) baseada no sistema de operação da máquina real. Basicamente, para o esquema de simulação foram utilizados os programas: *RSLogix 500*, leitura e edição do código *ladder, RSLogix Emulate 500*, simulação do controlador; e *Factorytalk View,* implementação da IHM. A comunicação entre as ferramentas citadas foi realizada por meio do programa *RSLinx*, o qual utiliza protocolos de comunicação padronizados na indústria. Este trabalho apresenta detalhes.

Palavras-chave: Automação industrial, CLP, IHM.

ABSTRACT

This paper presents the study and simulation of an industrial automation system for a filling machine in a food industry in the state of Ceará. The studied system's main component is a programmable logic controller (PLC) that runs a program developed in ladder language. The automation logic embedded in PLC was simulated by a group of softwares which allows the study and testing without the need for connection to the actual automation system. Furthermore, for purposes of operating the simulated system in this work, a human-machine interface (HMI) based on the operating system of the real machine was developed. Basically, the simulation process consists of: reading and editing of the ladder code in the RSLogix 500 software, CLP simulation via RSLogix Emulate 500 software and system operation from a HMI implemented in FactoryTalk View software. The communication between said tools has been performed through *RSLinx* software, which uses communication protocols standardized in the industry.
Keywords: Industrial Automation, CLP, HMI.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CLP	Controlador Lógico Programável
CPU	Cental Processing Unit
Vdc	Voltage direct current
RAM	Random-access memory
EEPROM	Electrically Erasable Programmable Read-Only Memory
IEC	International Electrotechnical Commission
XIC	Examine if closed
XIO	Examine if open
OTE	Output energize
OTL	Output latch
TON	Timer on-delay
RTO	Retentive timer on-delay
TOF	Timer of delay
CTU	Count up
CTD	Count down
EN	Enable
DN	Done
TT	Timer timing
CAN	Controller Area Network
ODVA	Open DeviceNet Vendors Association
TR	Terminating Resistor
IHM	Interface homem-máquina
OPC	OLE for Process Control
DDE	Dynamic Data Exchange

SUMÁRIO

1 INTRODUÇÃO

A automação é, sem dúvidas, um suporte para o desenvolvimento de diversos setores principalmente o industrial. Em processos de produção em larga escala, a automação desempenha um dos papeis principais para o aumento da eficiência. A necessidade crescente da humanidade para processos mais rápidos e produtivos, tornou o mercado da automação estável e abrangente. A área de aplicação é vasta, contemplando projetos de instalações elétricas, monitoramento, controle e proteção de sistemas industriais.

A automação industrial requer a aplicação de diversas áreas de conhecimento tal como, eletrônica, controle de sistemas, mecânica, física, química, comunicações e desenvolvimento de *software*. O sucesso desta solução fez com que ela se desenvolvesse em uma vasta área de aplicações. Atualmente ela engloba atuadores e sensores eletrônicos e mecânicos, um grande número de controladores lógicos programáveis (CLP), interfaces de operação, diversos protocolos de comunicação para todos os níveis de processo, servidores e sistemas supervisórios completos.

A utilização de um CLP em um projeto de automação possibilita a implementação de lógicas de controle através de programação computacional, não sendo necessária a inclusão e remoção de componentes físicos. Diversos fabricantes apresentam ótimas soluções em termos de controladores para os mais diferentes graus de complexidade de processo. Dependendo, da distância do controle para o ponto de operação e do número de variáveis controladas, pode ser necessária a utilização de redes de comunicação. A necessidade de uma rede de automação, torna o desenvolvimento do projeto ainda mais complexo, pois são diversos protocolos para serem escolhidos entre redes abertas e proprietárias.

Além dos CLPs, os sistemas de automação normalmente precisam da interação humana para operação e manutenção do sistema. Neste ponto, outro fator do projeto deve ser definido e implementado: o modo como a operação é controlada. Para esta etapa, há diversas soluções que vão desde simples botoeiras até complexos sistemas de monitoramento e análise a distância.

Dentre as ferramentas de *software* e *hardware* para desenvolvimento de sistemas de automação, os programas de simulação computacional estão sendo cada vez mais utilizados. A principal utilidade de tais ferramentas é a possibilidade de estudo e realização de testes sem a necessidade de um sistema físico real. Dessa forma, o engenheiro pode depurar a lógica de automação de forma mais segura e eficiente.

Dentro deste contexto, este trabalho apresenta o estudo e simulação de um sistema de automação de envase de tonéis de margarina através de *softwares* de padrão industrial. O estudo de caso investigado é baseado em um sistema real de uma indústria instalada no Estado do Ceará, Brasil.

1.1 Justificativa do trabalho

Como dito anteriormente, a área de automação industrial é fundamental no atual cenário de desenvolvimento do mercado de consumo e da indústria. Portanto, para um engenheiro eletricista, o conhecimento sobre o desenvolvimento de sistemas de automação industrial é de valiosa importância para o sucesso profissional.

1.2 Objetivos do trabalho

Este trabalho tem como objetivo principal a aplicação de um conjunto de *softwares* para a simulação e testes de um sistema de automação industrial. Dentre os objetivos específicos destaca-se o desenvolvimento de uma interface homem-máquina.

1.3 Composição do trabalho

Este trabalho está dividido em sete capítulos. Após a introdução do assunto, é apresentado um capítulo que aborda um dos componentes principais de um projeto de automação: o controlador lógico programável. Neste capítulo, os principais componentes e modo de funcionamento do controlador serão apresentados. Em seguida, ainda no mesmo capítulo, uma abordagem bastante detalhada da principal linguagem de programação (ladder) para controladores lógicos é feita.

Este trabalho foi feito baseado em procedimentos, adaptação de linguagem, softwares e componentes resultantes da parceria *Allen-Bradley/Rockwell Automation*. Esta escolha foi feita devido à popularidade das desenvolvedoras no mercado. Seguindo a sequência mencionada, no terceiro capítulo, são apresentadas as ferramentas computacionais utilizadas no trabalho.

No quarto capítulo, é realizada a apresentação do *hardware* de um caso real que é abordado em dois capítulos. O quinto capítulo prossegue com a abordagem do caso prático estudado. Neste capítulo, são explicadas as funcionalidades e particularidades do código *ladder* utilizado no caso real. Também foram abordados diversos passos de configuração que

devem ser realizados na implementação real de um projeto *ladder*, tentando recriar os procedimentos utilizados pelo programador original do projeto.

No sexto capítulo, é finalmente documentado todo o processo de simulação do caso em estudo e desenvolvimento da IHM utilizada para operar o sistema simulado. Este capítulo, apresenta os principais recursos que podem ser utilizados no desenvolvimento de uma interface. Além disso, são apresentados os resultados obtidos com a simulação da IHM desenvolvida.

Por fim, apesar de cada capítulo conter uma conclusão especifica do seu tema abordado, o último capítulo apresenta uma conclusão geral do trabalho, destacando as dificuldades enfrentadas, pontos positivos e negativos e sugestões para trabalhos futuros.

2 CONTROLADORES LÓGICOS PROGRAMÁVEIS

Os controladores lógicos programáveis são dispositivos amplamente utilizados em plantas industriais que desempenham funções de controle de processos. A principal diferença entre um CLP e um computador pessoal é a robustez. Esses controladores são projetados para suportar altas variações de temperatura, ruídos elétricos, impactos mecânicos e vibração, isto é, características encontradas em ambientes industriais e impróprias para o bom funcionamento de um computador comum. Antes de desenvolvimento do CLP, o controle de processos industriais era feito por painéis de relé. Com esta tecnologia, qualquer alteração na lógica de um processo significava uma grande perda temporal e econômica. Esta dificuldade era constantemente enfrentada pela indústria automobilística que necessitava de uma linha de produção altamente automatizada. Diante desse contexto, o CLP foi desenvolvido em 1968 pela *General Motors*, possibilitando maior flexibilidade e agilidade na alteração do controle de processos industriais [2].

Em comparação com o controle lógico realizado por painéis de relés, os CLPs apresentam diversas vantagens:

a) Flexibilidade: alterar a lógica de um processo é muito mais simples em um CLP, pois essa mudança é feita por meio de programação. Já em um painel de relés, uma alteração lógica necessitaria de um recabeamento entre os componentes do painel e, provavelmente, novos equipamentos teriam que ser adquiridos.
b) Custo: dependendo da complexidade do processo a ser automatizado, a instalação de sistema de CLP representa um menor investimento em comparação com painéis de relés. Isso é justificado pelo menor número de componentes necessários para um sistema de controlador lógico. Também, como já foi citado, futuras alterações representam um custo reduzido ou inexistente em sistemas com CLPs.
c) Comunicação: os CLPs possibilitam a comunicação com dispositivos de campo (transmissores, sensores, interfaces homem-máquina e etc.) e computadores por meio de diversos protocolos. Essa característica possibilita o supervisionamento do processo e operação remota.

2.1 Componentes de um CLP

Um CLP é composto basicamente, por uma unidade de processamento (CPU), memória, fonte e seção de entradas e saídas. Esses componentes podem estar todos presentes em um único bloco, caracterizando um CLP compacto, ou em módulos separados, caracterizando um CLP Modular.

2.1.1 Fonte

Esse componente é responsável pela conversão da tensão alternada para uma tensão contínua (geralmente utiliza-se 5 Vdc ou 24 Vdc). A fonte também garante uma operação normal do CLP diante de flutuações de tensão, característica fundamental em um ambiente industrial (grandes cargas sendo acionadas).

2.1.2 CPU

A CPU é responsável pela execução do programa, atualização de saídas, alocação de memória, entre outras funções. Em sistemas críticos, onde não pode haver falhas, pode ser usada uma CPU com redundância, isto é, o módulo de processamento pode possuir mais de um microprocessador e, em caso de falha do processador principal, o segundo assume o controle do sistema. O módulo da CPU geralmente possui LEDs indicadores de estado e uma chave seletora do modo de operação:

a) Modo *RUN*: o programa gravado no controlador é executado. Nesse modo, não é possível a edição da lógica por meio de um dispositivo de programação (geralmente, um computador pessoal)
b) Modo *PROG*: é possibilitada a edição do programa com segurança, pois todas as saídas são desenergizadas e não é possível trocar o modo de operação do CLP remotamente.
c) Modo *REM*: é possibilitada a troca do modo de operação do CLP por meio de um dispositivo de programação. Também é possível modificar o programa do controlador de forma *online*, isto é, a lógica da automação pode ser modificada durante a execução do projeto. A principal vantagem da edição online é evitar a perda dos estados das saídas, pois, em uma reprogramação *off-line*, esses dados são perdidos e é necessário um *startup* do processo controlado.

2.1.3 Memória

Responsável pelo armazenamento do programa do usuário e dos dados utilizados pelo programa. Esse item é fundamental na escolha do CLP ideal para a automação do processo, pois, de acordo com a complexidade do sistema, será definida a memória necessária e, consequentemente, o modelo de CPU a ser adquirido. A memória do CLP é especificada de acordo com a "palavra" utilizada pela CPU. Geralmente, os CLPs atuais trabalham com palavras de 16 bits, 32 bits ou 64 bits. Portanto, uma memória de 6 K indica uma capacidade de armazenamento de seis mil palavras.

A memória do CLP pode ser classificada em dois tipos: volátil ou não-volátil. A memória volátil perde os dados armazenado quando há uma queda ou perda total da alimentação do sistema. Esse tipo de memória é tipicamente utilizado para armazenamento das imagens de estado das entradas e saídas e dados das instruções do programa (temporizadores, contadores e etc.). Um exemplo de memória volátil é a *Random Access Memory* (RAM) que é utilizada quando há necessidade de leitura e escrita de dados de forma rápida. Alguns processadores são acompanhados de baterias para garantir uma autonomia de alguns minutos para a memória RAM. Para o armazenamento do programa e de certos parâmetros, é utilizada a *Electrically Erasable Programmable Read-Only Memory* (EEPROM). Essa memória é não-volátil e permite o armazenamento do programa do usuário mesmo em situações de queda de energia. Além disso, ela possibilita a edição dos dados armazenados por meio de um dispositivo de programação.

2.1.4 Módulos de Entrada/Saída

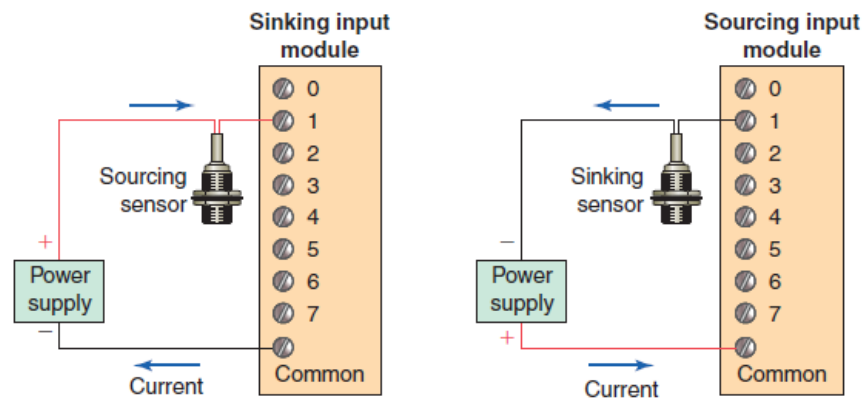
O módulo de entrada/saída é o componente do CLP responsável pela conexão dos dispositivos de campo com a unidade de processamento. Esta conexão é feita por meio de isolamento óptico, isto é, as tensões e correntes utilizadas por atuadores e sensores são convertidas por acopladores ópticos para o nível de tensão utilizada pelos equipamentos lógicos. O isolamento garante uma integridade para o controlador no caso de falta no circuito do dispositivo de campo que muitas vezes se encontra em um ambiente adverso. Além da proteção garantida pelo acoplamento óptico, a utilização de borneiras com fusíveis é muito comum na interligação dos módulos de entrada/saída com os dispositivos de campo. Geralmente, este módulo possui LEDs indicadores de estado de cada entrada ou saída para facilitar a manutenção do sistema.

Figura 2.1 – Módulos de um sistema SLC 500 da Allen-Bradley

Fonte: Folheto SLC 500 Programmable Controllers [1].

Os módulos de entrada/saída possuem diversas especificações. A mais básica é o tipo de sinal controlado: discreto ou analógico. Nos módulos de saída discretos, deve ser observado o tipo de comutador do módulo, os quais normalmente são do tipo relé, transistor ou TRIAC [2].

Os módulos de entrada/saída discretos também podem ser classificados como consumidores (*sinking*) ou fornecedores (*sourcing*) de corrente. Essa característica está relacionada com o fluxo de corrente adequado para o funcionamento do circuito de entrada ou saída. De acordo com a especificação do módulo e dos dispositivos de campo, é definida a polaridade da fonte contínua que deve ser ligada ao terminal comum do módulo do controlador: módulos consumidores de corrente têm seu terminal comum conectado com a polaridade negativa da fonte e os fornecedores possuem comum conectado a polaridade positiva da fonte.

Figura 2.2 – Ligações de entradas sink source

Fonte: Petruzella (2010, p. 26).

Fora as especificações já citadas para módulos discretos, os seguintes itens também são fundamentais na escolha do módulo adequado para o processo:

a) Tensão de entrada: especifica qual a magnitude e tipo de tensão a ser aplicada no módulo de entrada
b) Número de entrada/saídas: determina o número máximo de dispositivos que podem ser ligados ao módulo. Módulos de “alta densidade” podem possuir 32 ou 64 entradas/saídas. Esse módulo possui a vantagem de agrupar vários dispositivos em um único slot, mas a corrente por circuito é bastante reduzida. [2]
c) Tempo de resposta: especifica o intervalo de tempo que o módulo leva para detectar uma mudança de estado de uma entrada
d) Proteção contra curto-circuito: este item especifica que tipo de proteção é utilizada no módulo se saída. Também é mencionado se a proteção é exclusiva de cada saída ou compartilha entre grupos de saídas.
e) Corrente nominal do módulo: este valor determina a corrente consumida pelo módulo e é importante para o dimensionamento da fonte que alimenta o *rack* do CLP.

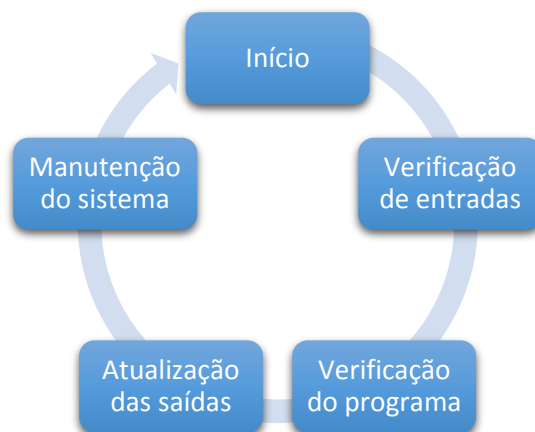
Os módulos analógicos são utilizados para sensores e atuadores de grandezas como temperatura, nível, pressão, vazão, carga, entre outras. O módulo pode ser sensível à corrente ou à tensão. Valores comumente encontrados nas amplitudes dos sinais de entrada/saída desses blocos são: 0 a 10 Vdc e 4 mA a 20 mA. Outro fator importante na especificação do módulo é a resolução que define a precisão e a sensibilidade da medição dos

sinais. O número e o tipo de ligação dos canais também devem ser especificados. Os canais podem possuir um comum para todas as entradas ou cada canal pode possuir um comum exclusivo. Canais diferenciais (com comum exclusivo) possuem maior precisão, pois esse tipo de ligação é mais eficaz contra ruídos eletromagnéticos.

2.2 Execução do Programa

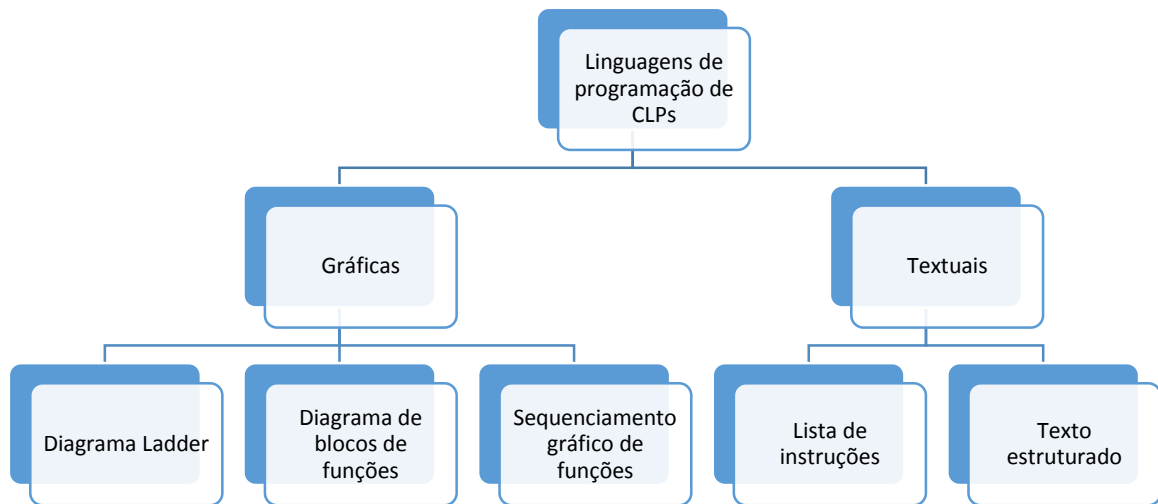
A execução do programa de um CLP pode ser, basicamente, dividida em quatro etapas: verificação de entradas, verificação do programa, atualização das saídas e manutenção do sistema. A execução dessas quatro etapas é denominada de ciclo de scan e o tempo que esse ciclo é executado é um fator determinante para o controle adequado do processo. Geralmente, o tempo de scan varia entre 1 ms e 20 ms [2]. Existem diversos fatores que influenciam esse tempo de scan, sendo o primeiro modelo de processador utilizado. O tempo de execução também aumenta de acordo com o número de instruções presentes no programa. A complexidade das funções executadas também influencia esse valor. O estado das condições lógicas também pode significar uma variação no tempo de execução de um ciclo de scan, pois o estado lógico dos itens em análise de cada linha pode definir a verificação do próximo item ou a passagem para a próxima linha do programa.

A Figura 2.3 ilustra como o ciclo de um CLP é executado. Primeiramente, o estado das entradas do controlador é verificado e copiado para um arquivo “imagem”. De acordo com o estado das entradas, o programa é executado e as decisões lógicas tomadas. Após a execução do programa, o arquivo de dados correspondente às saídas é atualizado e seus respectivos circuitos modificados. Entre cada ciclo, o controlador realiza uma verificação interna de seus componentes, sendo essa etapa denominada como manutenção do sistema.

Figura 2.3 – Fluxograma do ciclo de scan de um CLP

Fonte: Ilustração própria.

2.3 Linguagem de programação

Atualmente, as linguagens de programação de CLPs são padronizadas pela IEC 61131-3 99[2], sendo duas linguagens textuais e três linguagens gráficas como mostra a Figura 2.4.

Figura 2.4 – Diagrama com as linguagens de programação de CLPs

Fonte: Ilustração própria.

A linguagem ladder é a mais utilizada na programação de CLPs. Essa linguagem gráfica é muito semelhante aos diagramas de comandos elétricos por relés. Essa semelhança caracteriza o intuito de facilitar a transição da automação por relés para a automação por controladores programáveis. Além disso, por representar condições lógicas por meio de contatos, essa linguagem facilita muito o acompanhamento do funcionamento do código, garantindo uma rápida e prática manutenção do sistema. Devido a todas as vantagens citadas acima, o diagrama ladder será a linguagem abordada neste trabalho.

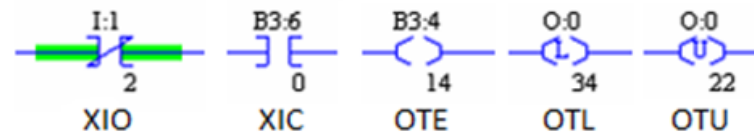
A linguagem ladder é composta por duas linhas de energia virtuais. O “polo positivo” ou fase fica a esquerda do esquema e a linha de energia que simula o ponto comum ou negativo fica a direita. Entre as linhas de energia do diagrama ladder, existe a zona de teste e a zona de ação. A zona de teste é composta por elementos que avaliam condições lógicas (contatos, blocos de comparação e etc.), permitindo ou não a passagem de uma “corrente virtual” em direção à zona de ação. Na zona de ação, são inseridos blocos de funções ou bobinas que desencadeiam ações de acordo com a energização da linha em que estão inseridos.

2.3.1 Principais funções da linguagem ladder

Na versão do *software* RSLogix 500 utilizado no desenvolvimento desse trabalho, estão disponíveis 127 funções da linguagem ladder. Nesta seção, serão apresentadas as funções básicas para o desenvolvimento de um projeto de acordo com as características do *software* citado acima

2.3.1.1 Instruções baseadas em relés

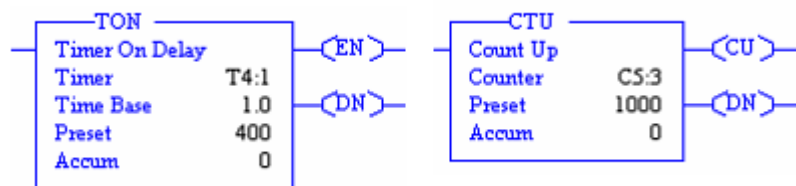
As funções baseadas em relés são basicamente 5: *examine if open* (XIO), *examine if closed* (XIC), *output energize* (OTE), *output latch* (OTL) e *output unlatch* (OTU). A Figura 2.5 demonstra graficamente essas funções:

Figura 2.5 – Representações das funções baseadas em relés no *software* RSLogix 500

Fonte: Ilustração própria obtida com auxílio do *software* RSLogix 500

Essas funções são associadas a bits e podem avaliar ou definir um estado. A função XIC, também conhecida como contato normalmente aberto, permite a energização da linha se o bit endereçado possuir nível alto. Já a função XIO ou contato normalmente fechado permite a energização da linha se o bit associado possuir nível baixo, bloqueando na situação contrária. As funções de output definem o estado do bit endereçado de acordo com a situação da linha, isto é, se a linha estiver energizada a respectiva saída será modificada para o nível 1 e, no caso contrário, a saída assumirá valor 0. As funções OTE e OTL se diferenciam no fator retenção, pois a OTL, após ser energizada, permanece no estado alto mesmo que a linha volte a ser descontinuada e a OTE dependerá apenas da continuidade atual da linha. A função OTU é utilizada para resetar um bit que foi acionado pela função OTL, isto é, quando for detectada continuidade na linha de uma função OTU, o bit associado a esta função será alterado para nível 0 mesmo que esteja em estado alto com retenção.

2.3.1.2 Contadores e Temporizadores

A categoria contadores e temporizadores pode ser representada basicamente por 5 funções: *timer on-delay* (TON), *timer off delay* (TOF), *retentive timer on-delay* (RTO), *count up* (CTU) e *count down* (CTD). A Figura 2.6 ilustra o bloco e os parâmetros base das funções TON e CTU.

Figura 2.6 – Representações das funções TON e CTU no *software* RSLogix 500

Fonte: Ilustração própria obtida com auxílio do *software* RSLogix 500

Temporizadores e contadores possuem dois parâmetros comuns: *acumulated value* (ACC) e *preset value* (PRE). No temporizador, o valor acumulado representa o número de intervalos base que foram contados e, no contador, o ACC registra o número de transições de subida ou descida que ocorreram na linha. O preset é um valor alvo para estas funções, isto é, quando o valor acumulado for maior ou igual ao preset, o bit *done* (DN) é ativado. O parâmetro *time base* determina a duração, em segundos, de um intervalo base.

No *software* RSLogix 500, por padrão, os temporizadores são indexados em um arquivo T4. Cada elemento temporizador possui 3 palavras de 16 bits com parâmetros de configuração, sendo a segunda palavra equivalente ao valor PRE e a terceira ao valor ACC. Na primeira palavra, estão disponíveis três bits para o usuário: *enable* (EN), *timer-timing* (TT) e o DN. O bit EN indica que a linha do temporizador é verdadeira. O bit TT indica que o temporizador está contando, isto é, o valor acumulado está sendo modificado e o bit DN não foi setado. Já o bit DN indica que o valor de preset foi alcançado. Esses três bits podem ser associados a instruções XIC ou XIO para manipulação de saídas.

As funções TON e TOF não possuem retenção, fazendo com que a palavra ACC seja resetada quando a condição da linha não atinge os requisitos para temporização. Na função TON a temporização é iniciada quando a condição da linha do temporizador é modificada para nível alto. Já na função TOF a contagem do valor acumulado é iniciada em uma transição de nível alto para nível baixo. A função RTO inicia a temporização de um evento quando a linha estiver energizada. Porém, quando a o nível da linha retorna para o valor 0, o temporizador não é resetado e o valor acumulado permanece o mesmo.

Os contadores de baixa frequência, assim como os temporizadores, possuem 3 palavras de 16 bits de dados e configurações, sendo a segunda palavra responsável pelo armazenamento do valor de preset e a terceira pelo valor acumulado. Na primeira palavra de um contador, são utilizados pela função CTU os seguintes bits: *count up overflow* (OV), DN e *count up enable* (CU). O bit CU é ativado sempre que a condição da linha do contador for verdadeira. O bit OV é setado quando o valor acumulado do contador ultrapassa o valor máximo de armazenamento (32767), recomeçando a contagem. O bit DN é utilizado nas funções CTU e CTD e indica que o valor acumulado alcançou o valor de preset. A função CTD utiliza os bits *count down* (CD) e *underflow* (UD) da primeira palavra de configuração do contador. O bit CD é dependente da condição lógica da linha do contador, isto é, se a linha for verdadeira, o bit estará em nível lógico alto. O bit UD indica que o valor acumulado foi decrementado além do limite de armazenamento para valores negativos (-32768), recomeçando a contagem decrescente do valor máximo de armazenamento.

Os contadores monitoram as transições da linha do valor falso para o valor verdadeiro. A função CTU faz um incremento do valor acumulado a cada borda de subida e a função CTD faz o decremento do valor acumulado na mesma situação. O contador mudará o estado do bit DN quando o valor acumulado for igual ou maior que o valor de preset. Mesmo após o valor de preset ter sido alcançado, o valor acumulado será alterado de acordo com as transições da linha do contador. As funções CTD e CTU podem ser associadas a um mesmo elemento contador, isto é, um contador pode ter um aumento e redução do valor acumulado em um mesmo processo. Um exemplo de aplicação para esta última situação é a automação de um estacionamento onde um sensor de entrada acionaria a função CTU, um sensor de saída estaria associado a uma função CTD e o preset do contador seria o número máximo de vagas do estacionamento.

Outra função que é comumente utilizada com temporizadores e contadores é a função reset (RES). Esta função reseta os bits de controle e o valor acumulado de contadores e

temporizadores. Para o funcionamento adequado da função RES, ela deve estar endereçada ao mesmo contador ou temporizador a ser resetado.

2.3.1.3 Instruções Matemáticas

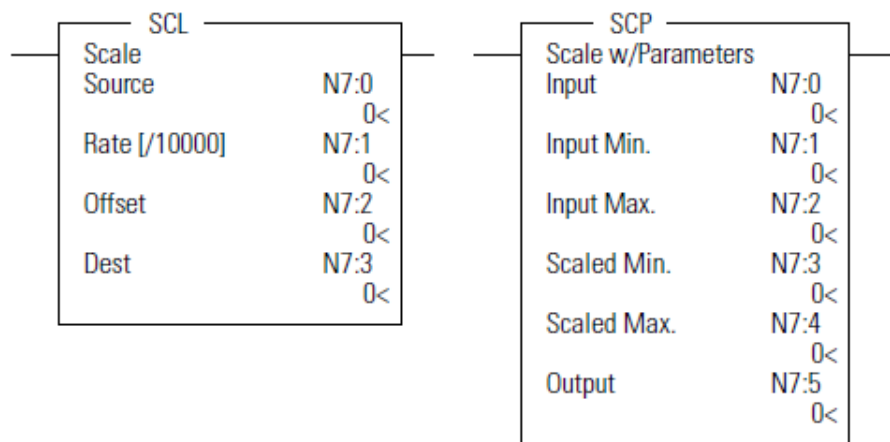
A maioria das funções matemáticas utiliza três parâmetros: *Source A*, *Source B* e *Destination*. A operação matemática é realizada utilizando os dois valores fontes (*source A* e *B*) e o resultado é armazenado no endereço destino (*Destination*). Os valores fontes podem ser constantes ou endereços, isto é, um valor pode ser declarado diretamente (constante) ou um endereço que contenha o valor a ser manipulado pode ser associado a uma fonte. Apesar desta flexibilidade, os dois valores fontes não podem ser duas constantes. A operação matemática será realizada quando a linha da função matemática for verdadeira. No caso do resultado da operação for maior que o valor máximo de armazenamento do endereço destino, o resultado pode ser definido de duas formas de acordo com o *Math Overflow Selection Bit*. Quando este bit for igual a zero e o resultado maior que o limite de armazenamento, o valor destino será um valor “saturado”, isto é, assumirá um valor limite do destino de acordo com o tamanho da palavra do endereço destino (-32678 ou 32767 para uma palavra de 16 bits). No caso do bit de seleção de *overflow* igual a 1, o valor destino receberá os 16 bits menos significantes do resultado da operação. A Tabela 2.1 mostra um resumo das funções matemáticas.

Tabela 2.1 – Resumo das principais funções matemáticas de um controlador Allen-Bradley

Sigla da Função	Nome (Inglês)	Descrição
ADD	Add	Adiciona dois valores fontes A e B e guarda o resultado em um endereço destino
SUB	Subtract	Subtrai da fonte A o valor da fonte B e salva em um destino
MUL	Multiply	Multiplica dois valores fonte
DIV	Divide	Divide o valor fonte A pelo valor fonte B
NEG	Negate	Troca o sinal do valor fonte
CLR	Clear	Zera o valor destino
SQR	Square Root	Calcula a raiz quadrada do valor fonte
SCL	Scale	Calcula o valor correspondente em uma nova escala do valor fonte
SCP	Scale with parameters	Calcula o valor correspondente em uma nova escala por meio de uma relação linear

Fonte: Tabela própria.

Das funções apresentadas na Tabela 2.1 é válido destacar as funções ADD, SCL e SCP. A função da operação de adição, apesar da simplicidade, é muito utilizada na contagem de ciclos de produção, controle de valores inteiros e instruções de comparação de valores numéricos. As funções de mudança de escala (SCL e SCP) são fundamentais na implementação de lógica com leituras analógicas, pois é possível converter rapidamente um valor analógico com escala de 4 mA à 20 mA para a escala de uma grandeza do processo como peso ou temperatura.

Figura 2.7 – Representações das funções SCL e SCP

Fonte: Manual do controlador Micrologix 1500 com adaptações [3].

Observando a Figura 2.7, é possível perceber a diferença básica entre essas duas funções de mudança de escala. A função SCP é mais amigável ao usuário, pois a conversão é feita automaticamente após o fornecimento dos valores limites das escalas. Já a função SCL requer que o usuário calcule um fator multiplicativo (*rate*) e um *offset* para realizar a mesma conversão. As equações (1), (2) e (3) mostram o cálculo realizado para conversão de escalas de acordo com as variáveis apresentadas nos blocos da Figura 2.7.

$$\text{Dest} = \frac{\text{Source} \cdot \text{Rate}}{1000} + \text{offset} \qquad (1)$$

$$\text{Rate} = \frac{\text{Valor de escala}_{\text{Máximo}} - \text{Valor de escala}_{\text{Mínimo}}}{\text{Valor de entrada}_{\text{Máximo}} - \text{Valor de entrada}_{\text{Mínimo}}} \qquad (2)$$

$$\text{offset} = \text{Valor de escala}_{\text{Mínimo}} - \text{Valor de entrada}_{\text{Mínimo}} \cdot \text{Rate} \qquad (3)$$

Apesar da facilidade da função SCP, a função SCL apresenta a vantagem de, no caso do fornecimento dos parâmetros por endereçamento, utilizar menos espaço para armazenar os parâmetros. Outra vantagem é o tempo de execução da função SCL que chega a ser 3 vezes menor que o tempo de execução da função SCP [3].

2.3.1.4 Instruções de comparação

As instruções de comparação utilizam dois valores (*Source A* e *Source B*) para definir a continuidade de uma linha lógica. Essas funções podem ser comparadas com as funções do tipo relé em um aspecto, pois as duas são utilizadas para testar os valores de entrada e definir uma ação a ser feita no fim da linha lógica, sendo que as instruções do tipo relé testam bits de entrada e as instruções de comparação testam palavras de dados. A Tabela 2.2 exibe as principais funções de comparação e suas respectivas expressões lógicas com base nos parâmetros utilizados em controladores do fabricante Allen-Bradley.

Tabela 2.2 – Resumo das principais funções de comparação de um controlador Allen-Bradley

Sigla da Função	**Nome (Inglês)**	**Descrição**	**Expressão Lógica**
EQU	Equal	Testa se dois valores são iguais	A=B – Verdade
NEQ	Not equal	Testa se dois valores são diferentes	A≠B – Verdade
LES	Less than	Testa se um valor fonte é menor que o outro	A<B – Verdade
LEQ	Less than or equal	Testa se um valor fonte é menor ou igual ao outro	A≤B – Verdade
GRT	Greater than	Testa se um valor fonte é maior que o outro	A>B – Verdade
GEQ	Greater than or equal to	Testa se um valor fonte é maior ou igual ao outro	A≥B – Verdade
MEQ	Mask compare for equal	Testa se porções de dois valores são iguais	
LIM	Limite test	Testa se o valor fonte está dentro de um intervalor específico	

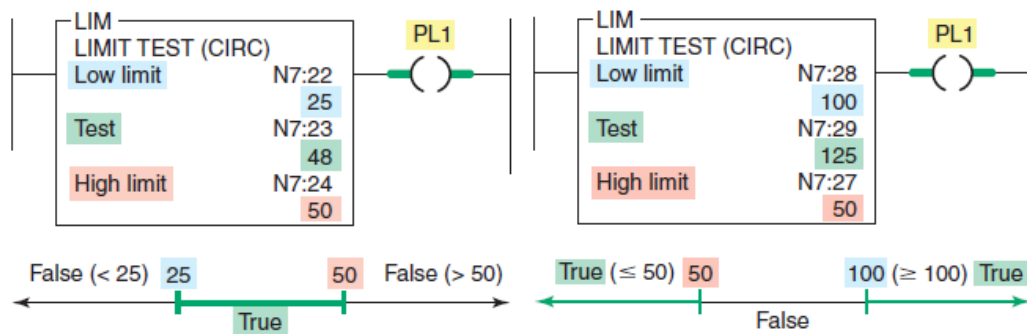
Fonte: Tabela própria com dados do manual do controlador Micrologix 1500

As seis primeiras funções da Tabela 2.2 utilizam apenas dois valores nas comparações e podem ser facilmente entendidas pelas descrição e expressão lógica fornecidas na tabela. Porém, as funções MEQ e LIM possuem diferentes parâmetros e uma expressão lógica de menor simplicidade.

A função MEQ permite que apenas alguns bits da palavra fonte sejam comparados com o valor referência e utiliza três parâmetros para definição da continuidade da linha:

source, *mask* e *compare*. A palavra *source* e a palavra *mask* são inicialmente submetidas a uma operação lógica AND, isto é, o usuário pode definir, por meio de uma “máscara”, quais bits da palavra *source* serão comparados. Em seguida, o resultado da operação AND é comparado com o valor indicado pela palavra *compare*. Todos os parâmetros dessa função devem possuir o mesmo tamanho (*word* ou *long-word*)

A função LIM utiliza três parâmetros: *low limit*, *test* e *high limit.* Quando o valor associado ao *low limit* for menor ou igual ao valor associado ao *high limit*, essa função retorna um valor verdadeiro se a palavra a ser testada estiver dentro do intervalo estabelecido pelos valores dos parâmetros limite inferior e superior. Quando o valor do parâmetro *low limit* for maior que o valor do *high limit*, a função retornará um valor verdadeiro se o valor a ser testado estiver fora do intervalo estabelecido pelos valores limites. A Figura 2.8 ilustra as duas possíveis situações referentes aos valores limite.

Figura 2.8 – Possíveis situações na utilização de um bloco da função LIM

Fonte: Petruzella (2010, p. 212) com adaptações.

2.3.1.5 Instruções de controle de programa

As funções de controle de programa são instruções que permitem a mudança da sequência de scan, possibilitando uma redução do tempo de execução do programa e a simulação de situações para corrigir problemas no ladder. A Tabela 2.3 mostra as principais funções de controle.

Tabela 2.3 – Resumo das principais funções de controle de um programa ladder.

Sigla da Função	**Nome (Inglês)**	**Descrição**
JMP	Jump to label	Pula para a linha com que possui a label associada
LBL	Label	
JSR	Jump to subroutine	Inicia a sub-rotina indicada e retorna
SBR	Subroutine label	
RET	Return from subroutine	
TND	Temporary end	Determina um fim prematuro do ciclo de scan
END	Program End	Determina o fim da execução do arquivo ladder associado
MCR	Master control reset	Define uma zona a não ser executada em todos os ciclos

Fonte: Tabela própria com dados do manual do controlador Micrologix 1500.

A função JMP deve ser utilizada em conjunto com a instrução LBL. Quando a condição da linha da instrução JMP for verdadeira, a ordem de scan do programa será modificada e o controlador irá verificar a linha indicada pela LBL associada com o mesmo endereço da função JMP. Todas as linhas ignoradas na execução da função JMP mão serão atualizadas e as entradas e saídas pertencentes a este intervalor permanecerão com seu último estado. Contadores e temporizadores não funcionam corretamente se estiverem dentro deste intervalo. Por isso, esses acumuladores devem ser programados fora da zona que pode ser ignorada na execução da função JMP.

A função JMP permite a execução de uma linha fora da ordem de scan. Porém, quando se quer executar várias linhas foram da ordem original do programa, são utilizadas as sub-rotinas. Na implementação de uma sub-rotina, as funções JSR, SBR e RET devem ser utilizadas em conjunto, sendo as duas primeiras associadas ao mesmo endereço. Se a linha de uma instrução JSR for verdadeira, o programa executará as instruções seguintes a *label* SBR associada. Ao executar a função RET (presente no fim de uma sub-rotina), o programa executará a próxima linha após a função de chamada da sub-rotina. Geralmente, as sub-rotinas são armazenadas em arquivos ladder separados do programa principal, permitindo uma maior organização do programa e evitando a execução desnecessárias de zonas de instruções

A instrução MCR é uma solução digital que simula a funcionalidade de um relé de segurança. Por meio de duas linhas com a instrução MCR, é definida uma zona que desenergizará todas as saídas não retentivas quando a condição lógica associada a primeira função MCR for falsa. Quando a linha da primeira instrução MCR for verdadeira, a zona delimitada pelas duas instruções de *máster control* serão ignoradas. Normalmente, são utilizadas faltas do controlador para executar uma zona MCR, permitindo a desenergização de uma parcela do programa em uma situação de emergência. Apesar de todas as funcionalidades

descritas, a instrução MCR não substitui um relé de segurança real para garantir a parada sem riscos de um equipamento.

As instruções TND e END determinam o fim de um ciclo de scan. Porém, a instrução TND é utilizada com alguma condição lógica atrelada para determinar um fim prematuro do ciclo de scan, permitindo que em determinadas situações uma parte do código não seja verificada e o tempo de execução seja manipulado de forma mais eficiente. Já a instrução END é utilizada no final de um programa ladder para indicar o recomeço do ciclo.

2.4 Conclusões

Nesta seção, foi apresentada uma abordagem sobre os principais componentes de um CLP. O conhecimento sobre as especificações do controlador e suas expansões é fundamental para iniciar-se um estudo sobre programação de CLPs e outros temas mais avançados como redes industriais e sistemas supervisórios. Além das especificações de *hardware*, foram abordados as principais instruções e o funcionamento da linguagem ladder que é a mais popular dentre as linguagens de controladores especificadas pela IEC 61131-3. O fato desta linguagem ser do tipo gráfica contribui para a popularização desta linguagem entre usuários que não estão habituados ao uso de linguagens de programação textuais de alto nível.

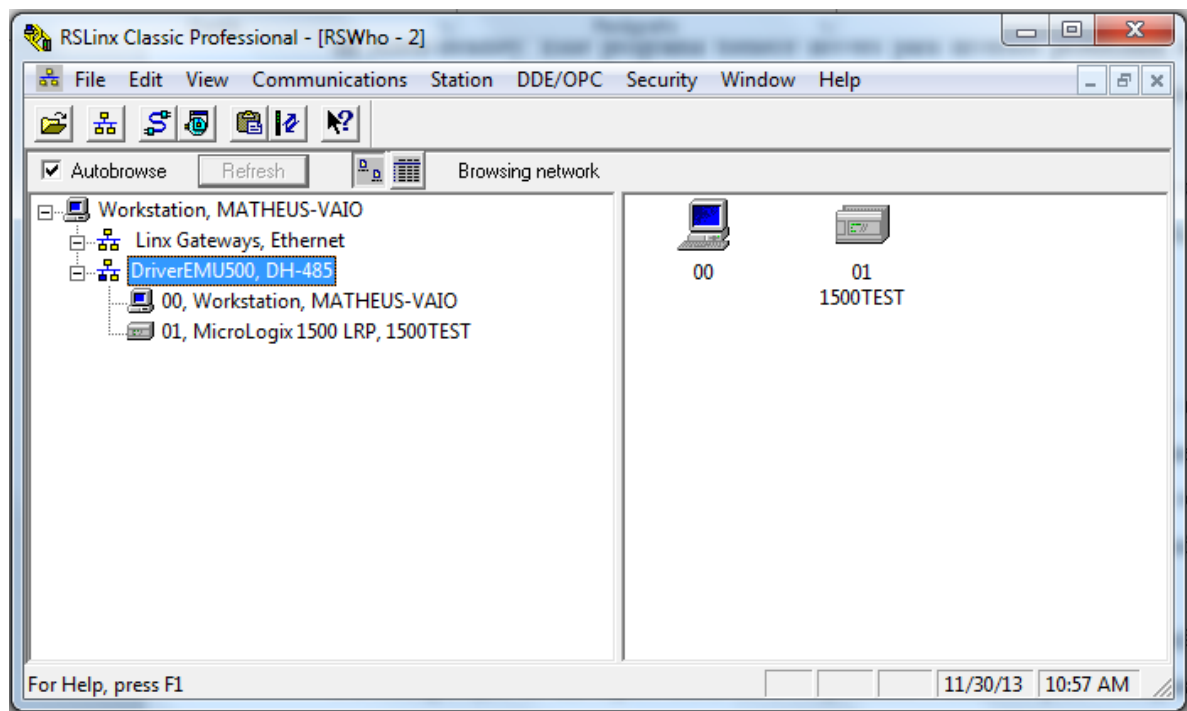
3 PROGRAMAS UTILIZADOS

O caso de automação estudado neste trabalho, utiliza componentes da fabricante *Allen-Bradley*. Por isso, nesta seção, serão discutidas as ferramentas computacionais da desenvolvedora *Rockwell Automation* que foram utilizadas no desenvolvimento deste trabalho. Serão apresentadas as interfaces e principais funções de cada *software* utilizado.

3.1 RSLinx

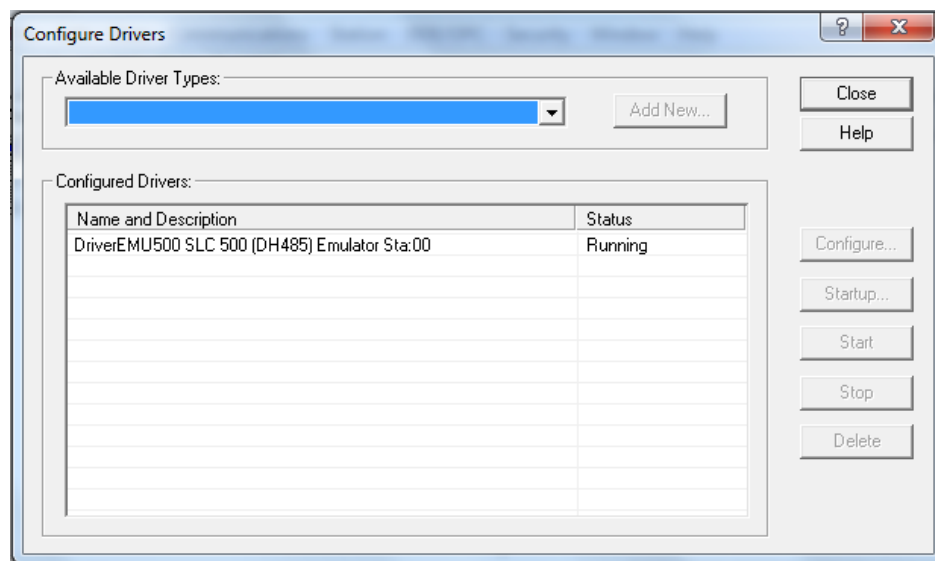
O *RSLinx* é um *software* desenvolvido pela *Rockwell Automation*. Esse aplicativo é responsável pelo estabelecimento da comunicação entre softwares da *Rockwell* e dispositivos da Allen-Bradley. Esse programa fornece drivers para diversos protocolos de comunicação utilizados por dispositivos de automação, possibilitando a leitura e escrita dos programas ladder por computadores pessoais.

A tela do *RSLinx* é bem simples e intuitiva, sendo composta por um menu, barra de ferramentas com as principais funcionalidades e a janela principal: RSWho. A Figura 3.1 ilustra a tela inicial do aplicativo.

Figura 3.1 – Tela Inicial do *software* RSLinx

Fonte: Ilustração própria obtida com auxílio do *software RSLinx* Classic

O RSWho apresenta as redes e dispositivos conectados de maneira muito semelhante ao Windows Explorer. Essa tela principal está dividida em duas janelas: a esquerda as redes e dispositivos estão dispostos “em árvore”, permitindo a expansão e recolhimento de grupos de acordo com a estação ou protocolo de comunicação e a direita o nó selecionado é exibido com mais detalhes.

As duas principais funções utilizadas no programa são a configuração de um driver e a configuração de um tópico *Dynamic Data Exchange/ OLE for Process Control* (DDE/OPC). A primeira é utilizada para configurar um driver de acordo com o protocolo de comunicação e o dispositivo que o usuário quer se comunicar. Na janela “*Configure* drivers” o usuário pode ver e controlar uma lista de drivers já configurados. Também é possível adicionar novos drivers por meio de uma lista de drivers disponíveis. Após a escolha do protocolo, o usuário deve escolher um nome para o driver e em uma segunda tela informar parâmetros específicos do protocolo escolhido. A Figura 3.2 mostra a janela de configuração de drivers.

Figura 3.2 – Configuração de drivers no *software RSLinx*

Fonte: Ilustração própria obtida com auxílio do *software RSLinx Classic*

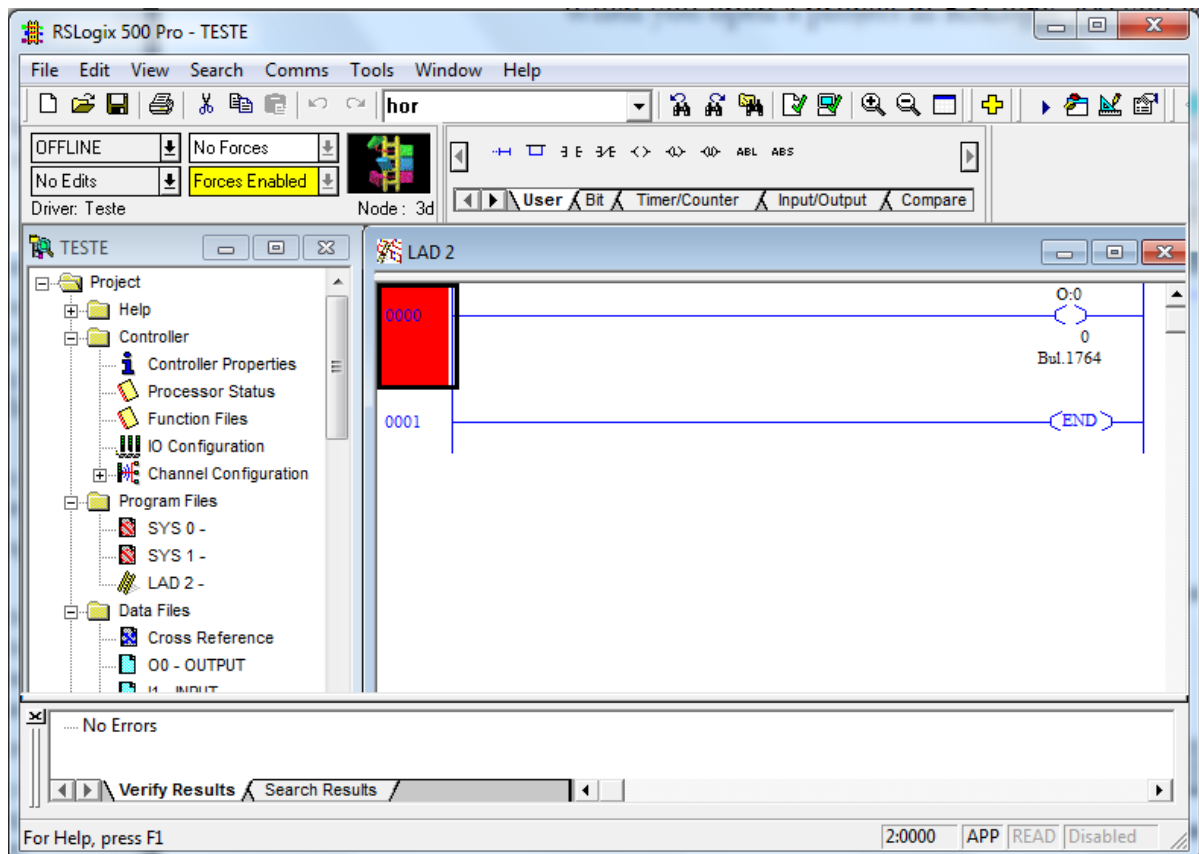
Já a segunda possibilita criar, copiar ou apagar um tópico DDE/OPC. *OLE for Process Control* é um padrão de comunicação desenvolvido pela OPC Foundadion, da qual muitas fabricantes de dispositivos de automação participam, que permite que dispositivos do chão de fábrica se comuniquem com aplicações do ambiente Windows. *Dynamic Data Exchange* é um protocolo de comunicação padrão entre aplicativos executados no sistema Windows. O DDE permite a troca de dados entre dois programas. A criação de um tópico

DDE/OPC no *RSLinx* permite que sistemas supervisórios acessem e modifiquem informações dos controladores lógicos.

3.2 RSLogix 500

O RSLogix 500 é o *software* utilizado para o desenvolvimento de programas ladder dos controladores SLC 500 e Micrologix da Allen-Bradley. A Rockwell Automation conta com outros dois compiladores ladder: RSLogix 5000, utilizado para programar CLPs ControlLogix, e o RSLogix 5 que é utilizado com os controladores PLC-5 da Allen-Bradley. Neste trabalho, o compilador utilizado foi o RSLogix 500.

A Figura 3.3 mostra a tela principal do *software* RSLogix 500.

Figura 3.3 – Tela principal do *software* RSLogix 500

Fonte: Ilustração própria obtida com auxílio do *software* RSLogix 500

A tela principal do programa conta com o menu de onde é possível acessar todas as funções e configurações do programa. Ainda na parte superior, há uma barra de ícones com acesso direto às principais funções edição, compilação e busca. Abaixo da barra de ícones,

temos uma caixa com informações de comunicação e a caixa de instruções. A caixa de comunicação exibe o modo de operação atual do controlador, a existência bits “forçados”, o driver utilizado e o nó do equipamento. Já a caixa de instruções disponibiliza os principais elementos do código ladder organizados em abas de acordo com suas funções. No canto esquerdo os componentes do projeto em desenvolvimento, como arquivos do controlador e módulos, programas ladder e arquivos de dados, são exibidos em uma disposição em árvore. Ao lado dos arquivos de projeto, está a tela de programação em ladder onde o código do controlador será desenvolvido. Na parte inferior da janela, está a divisória de resultados que mostra erros de compilação e ocorrências de buscas. O último componente da tela do *software* é a barra de status como mostra instruções de acordo com a operação que está sendo realizada.

O primeiro passo para começar a programar um CLP é a criação de um novo projeto. Durante a criação do projeto, diversas características, como nome do CLP, modelo do controlador, driver de comunicação utilizado e o número do nó do dispositivo. Porém, esses dados podem ser alterados a qualquer momento por meio do arquivo *Controller Properties* na árvore de projeto (podendo haver perda de dados). Após realizar essas configurações, o usuário deve terminar a especificação do *hardware* definindo, se for existente, os módulos de expansão e o rack utilizado. Essa configuração é feita por meio do arquivo de projeto *I/O Configuration*. Na janela de configuração de I/O há uma lista selecionável de cartões de expansão para o controlador especificado. Por meio do botão *Read I/O Config.*, essa configuração pode ser feita de maneira automática quando o controlador estiver conectado ao compilador.

Após a configuração dos componentes de *hardware*, é necessário entender como é feito o armazenamento de dados do controlador e como esses dados podem ser acessados. Os arquivos de dados possuem uma pasta na árvore de projetos e é a porção da memória do controlador que armazena o status das entradas, saídas, processador, bit virtuais, contadores, temporizadores e etc. Geralmente, o compilador RSLogix 500 possui 8 arquivos de dados pré-configurados divididos de acordo com o tipo de informação que o arquivo armazena:

a) O0 – *Output*: esse arquivo armazena os dados das saídas físicas do controlador
b) I1 – *Input*: arquivo que armazena os dados das entradas do sistema
c) S2 – *Status*: arquivo que armazena estados do controlador, como *flags* de operações matemáticas, modo de operação, *flags* de erros e etc.

d) B3 – *Binary*: armazena dados do tipo binário. São utilizados como entradas/saídas discretas virtuais na implementação do programa.

e) T4 – *Timer*: armazena os dados de temporizadores. Um único arquivo do tipo *timer* pode armazenar até 255 temporizadores, sendo cada temporizador composto por 3 palavras de dados (6 bytes).

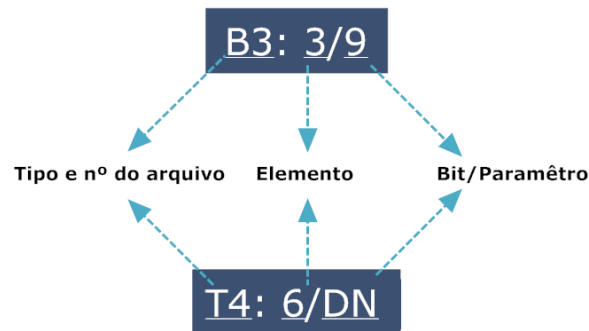
f) C5 – *Counter*: armazena o estado, valor acumulado e valor pré-selecionado dos elementos contadores. Assim como o arquivo de temporizadores, até 255 elementos podem ser armazenados em um único arquivo.

g) R6 – *Control*: armazena elementos de controle compostos por três palavra: uma de estado, uma para a posição de sequência e outra para tamanho da sequência. Palavra de controle são usadas por instruções como registradores de deslocamento e comparadores sequenciais.

h) N7 – *Integer*: este arquivo é utilizado para armazenamento de valores inteiros. Cada elemento possui 16 bits, podendo armazenar um número inteiro entre -32768 e 32767. Cada arquivo padrão do tipo inteiro pode guardar até 255 elementos.

De acordo com a lista acima, é possível perceber que cada arquivo possui uma sigla composta por uma letra e um número, representando o tipo de dado e o número do arquivo. Fora estes arquivos padrões, o usuário pode criar novos arquivos de dados de acordo com a necessidade da programação. Para criar um novo arquivo, o usuário deve especificar o número do arquivo, o tipo de dado e a quantidade de elementos. Opcionalmente, podem ser especificados nomes, descrições e restrições de acesso. Além dos tipos já citados, podem ser utilizados dados do tipo *float, string, long, message*, PID e *programmable limit switch*.

Os arquivos de dados podem ser acessados por meio de seus endereços ou "símbolos". O acesso por meio do endereço consiste em especificar o arquivo, elemento e bit/palavra da informação desejada. Já o acesso por símbolos consiste na chamada da informação por meio de uma *tag* previamente criada e associada a um endereço. A segunda alternativa deixa o desenvolvimento e soluções de problemas muito mais intuitivos e organizados. A Figura 3.4 ilustra como é composto um endereço no *software* RSLogix 500, mostrando que alguns elementos podem ter seus parâmetros referenciados por siglas.

Figura 3.4 – Endereçamento no *software* RSLogix 500

Fonte: Ilustração própria.

Após toda a configuração de *hardware* e *software*, pode-se iniciar o desenvolvimento do programa ladder. A programação pode ser feita simplesmente arrastando os componentes da caixa de instruções ou por meio de texto composto por sigla das instruções e endereços. A caixa de entrada para texto é exibida após um clique duplo na linha. Feita a lógica da linha, basta associar endereços às instruções.

Estando o código finalizado e o devido driver do controlador configurado no *RSLinx*, o usuário pode conectar o computador ao CLP e realizar o download do programa para o controlador. Após essa última etapa, os dados do CLP podem ser monitorados e os ajustes finais realizados. Nesta situação, o RSLogix ainda oferece diversas ferramentas como edição online, monitoramento de dados, backups automáticos e etc. A edição online permite a modificação do código sem a parada do controlador/processo e perda de estados. O usuário pode observar o comportamento do programa por meio dos arquivos *Data Monitors* e *Graphical Monitors*. O primeiro permite visualizar o estado de dados selecionados pelo o usuário e o segundo permite uma análise mais detalhada por meio de gráficos. Outra ferramenta bastante utilizada em programas extensos é a *Cross Reference* que permite encontrar outras ocorrências de um endereço selecionado dentro do programa.

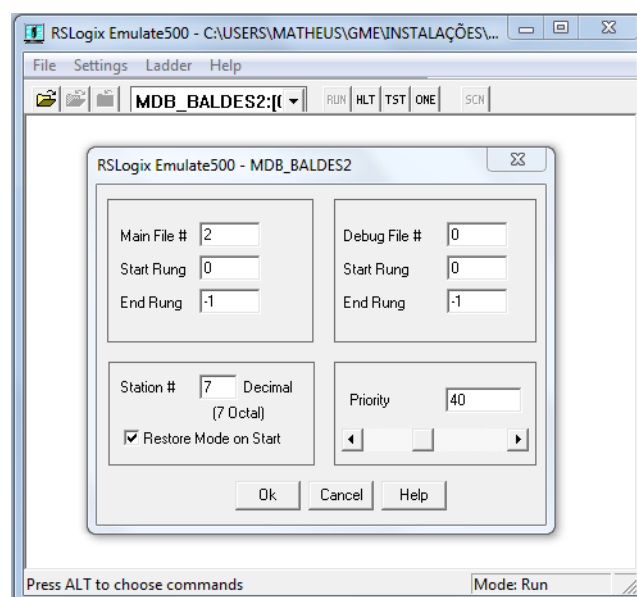
3.3 RSLogix Emulate 500

O RSLogix Emulate 500 é um simulador da família de controladores PLC-5 e SLC-500 da Allen-Bradley. Com este *software*, pode-se testar um projeto com as condições próximas da situação real que o controlador irá enfrentar. Observações de partes específicas do código desenvolvido e solução de problemas podem ser feitas com as facilidades oferecidas pelas simulações.

Dentre os softwares utilizados nesse trabalho, o programa de simulação de controladores Allen-Bradley é o mais simples. As informações exibidas na tela principal do programa são quase inexistentes e a configuração pode ser feita em um único passo.

A Figura 3.5 mostra os dois principais ambientes utilizados no Emulate 500: a tela principal e a janela de configuração do controlador simulado. A tela principal é composta apenas de uma barra de menu e uma barra de ícones. Na barra de ícones, há uma caixa de listagem com os processadores simulados e botões para seleção do modo de operação dos controladores. O *software* permite a seleção de 5 modos:

a) RUN: coloca o controlador virtual em operação.
b) HLT: o modo *halt* finaliza a simulação do controlador.
c) TST: o modo *Single File Scan* realiza uma execução de um arquivo ladder
d) ONE: o modo *One Rung Scan* realiza uma execução de uma linha lógica.
e) SCN: o modo *Single Scan* executa um ciclo de scan e para.

Figura 3.5 – Tela principal e janela de configuração do *software* RSLogix Emulate 500

Fonte: Ilustração própria obtida com auxílio do *software* RSLogix Emulate 500

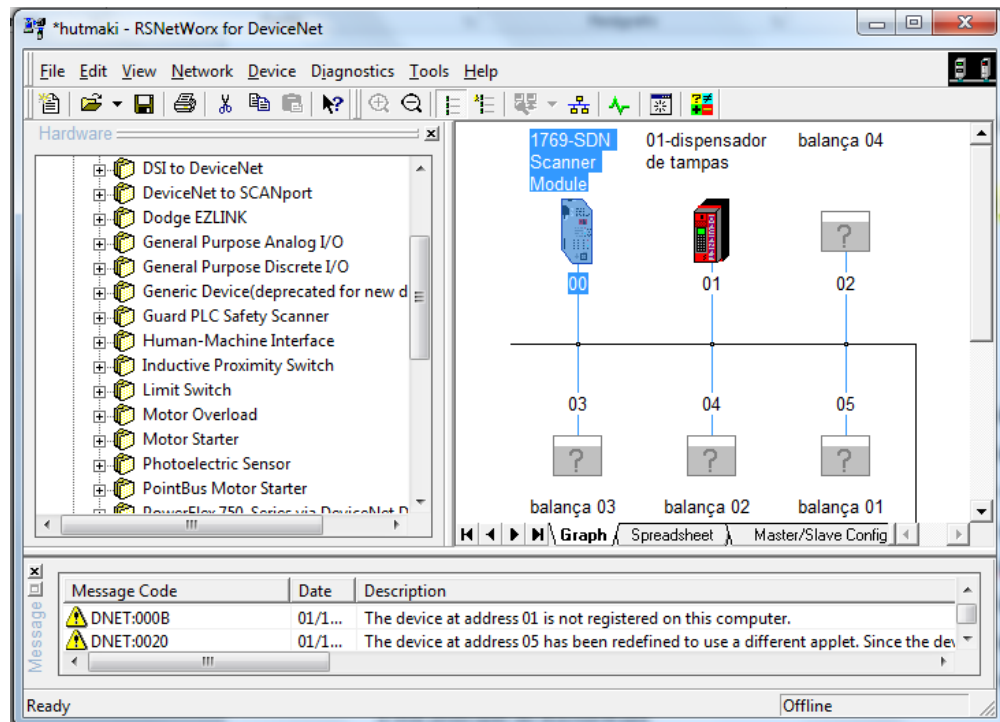
Para simular um processador, o usuário precisa configurar os driver do simulador no *software* de comunicação *RSLinx*. Controladores SLC-500 e Micrologix utilizam o driver de simulação “SLC-500 (DH485) Emulator”. Configurado o driver, basta abrir um projeto ladder existente e configurar como a simulação será feita. Na Figura 3.5, a tela de configuração é exibida. Ela permite a escolha do arquivo com o programa principal, escolha de um arquivo *debug*, primeira e última linha a ser verificada e o número da estação definido no *RSLinx*.

O RSLogix Emulate 500 não possui nenhuma interface para simulação e observação de entradas e saídas. Por isso, para simular o comportamento do programa de acordo com os elementos sensores e atuadores encontrados no processo, o usuário pode mudar as saídas no programa de edição ladder ou utilizar um arquivo de depuração. A primeira opção pode ser feita utilizando a função *toggle bit* do RSLogix que alterna o estado atual de entradas discretas. Essa opção é eficiente apenas para observar partes especificas do código. Para simular um estado mais próximo possível de um processo real, os arquivos de depuração devem ser utilizados. Esses arquivos são criados diretamente no programa de edição dos códigos ladder e não são transferidos para o processador quando um *download* é realizado. Estes arquivos são programados como qualquer outro arquivo ladder, permitindo o uso de temporizadores, contadores, transferência de dados e etc. Com isso, a execução de um processo pode ser simulada, fazendo com que os elementos sensores atuem de acordo com um padrão esperado na situação real de funcionamento da automação.

Durante a simulação, erros podem ser encontrados. Alguns apresentam um comportamento inesperado temporário que não permite a verificação do porquê da ocorrência da falha. Pensando nessa situação. O RSLogix Emulate 500 permite a configuração de *breakpoints* ou pontos de parada. Para utilizar os pontos de parada, basta o usuário escrever uma linha temporária no código que contenha as condições desejadas para executar a parada. Feito isso, o usuário pode configurar os pontos de parada no Emulate 500 por meio do menu “Ladder”. A configuração é feita especificando o número do arquivo do programa, linha, números de scans que devem ocorrer com a condição de parada sendo satisfeita e qual a condição da linha dispara a parada (Verdadeira, falsa ou qualquer). Todos esses parâmetros são referentes a linha temporária que o usuário criou no projeto ladder.

3.4 RSNetWorx

O *software* RSNetWorx da Rockwell Automation é utilizado para documentação e configuração de redes DeviceNet. Conforme ilustrado na Figura 3.6, a parte superior da janela principal do aplicativo é composta por uma barra de menus e uma barra de ícones com acesso direto a opções de edição e diagnóstico. No canto esquerdo, há uma lista de *hardware* disponíveis para serem adicionados a rede. A janela principal do programa é denominada *Configuration View* ou vista de configuração. Nesta janela, a rede DeviceNet pode ser visualizada graficamente, por tabela, ou pela relação mestre/escravo dos dispositivos. Na parte inferior da tela principal, é exibida a janela de mensagens, que mostra os eventos e estados dos dispositivos na rede.

Figura 3.6 – Tela principal do *software* RSNetWorx

Fonte: Ilustração própria obtida com auxílio do *software* RSNetWorx.

Para configurar uma rede DeviceNet pelo *software* RSNetWox, um novo arquivo deve ser criado no *software*. O segundo passo é o comissionamento dos componentes pertencentes a rede, isto é, os parâmetros básicos, como nó e baud rate, devem ser configurados nos próprios dispositivos, pois muitos dispositivos vem configurado com o mesmo nó padrão, impedindo que eles sejam lidos pelo RSNetWox.

Em seguida, com o driver de comunicação devidamente configurado, o usuário pode realizar um upload do estado atual da rede. Neste ponto, os dispositivos já podem ser configurados diretamente no *software*. Na janela propriedades de um dispositivo, podem ser feitas diversas configurações como endereço do arquivo, descrição, edição de parâmetros, dados de entrada e saída. É importante ressaltar que equipamentos que não pertencentes à parceria Rockwell/Allen-Bradley não estão na biblioteca de *hardware* do programa. Para configurar dispositivos de outros fabricantes, um arquivo "*Eletronic Data Sheet"* (EDS) do dispositivo deve ser fornecido.

Após a configuração dos escravos, o cartão *scanner* deve ser configurado. A configuração do *scanner* é realizada da mesma forma que dos dispositivos escravos, porém esta configuração é mais complexa. Os parâmetros como plataforma, nó da rede e slot no rack do CLP devem ser especificados. Uma lista de dispositivos a serem verificados e o tamanho dos dados de entrada e saída de cada escravo são informados em uma segunda aba de configuração. Os endereços de cada entrada e saída na tabela de dados do *scanner* pode ser conferido nas abas "*Inputs*" e "*Outputs*".

3.5 Conclusões

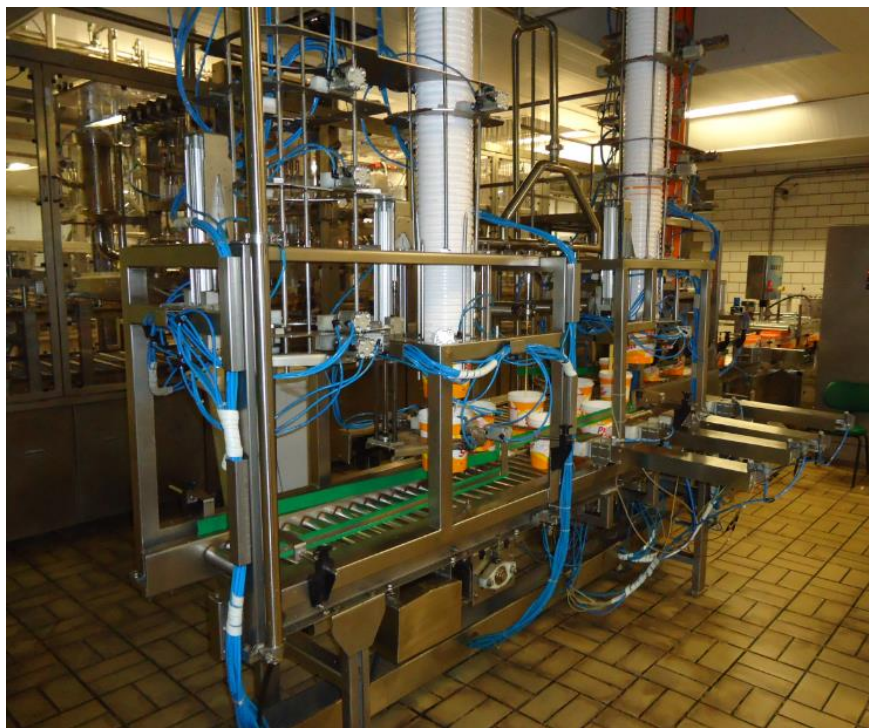
O terceiro capítulo é fundamental para o cumprimento do objetivo deste trabalho. Além do conhecimento do CLP e de como um código em ladder deve ser implementado, é necessário conhecer quais ferramentas são utilizadas para o desenvolvimento de cada tarefa de um projeto de automação. Contando apenas com softwares desenvolvidos pela Rockwell Automation, são utilizados diferentes programas para comunicação de dispositivos, simulação, programa em ladder, configuração de rede e desenvolvimento de interfaces gráficas. Inicialmente, a utilização dos softwares se mostrou bastante confusa, pois um mesmo fabricante utiliza vários programas, chegando a apresentar mais de duas ferramentas para a mesma tarefa. Apesar da popularização do conceito de interoperabilidade, com padronização de linguagens e protocolos de comunicação abertos, cada fabricante ainda apresenta sua própria solução para o comunicação e programação por exemplo. Isto, sem dúvidas, dificulta a linha de aprendizado do usuário já que tem que ser feita a troca de *software* para realizar outra tarefa e, para uma mesma tarefa, tem que ser feitas adaptações de um fabricante para outro.

4 HARDWARE DO CASO ESTUDADO

4.1 Componentes do sistema de automação do caso estudado

O sistema de automação considerado neste trabalho consiste em uma máquina envasadora/dosadora de baldes, que permite a dosagem e envase de uma linha de produção de margarina. De forma simplificada, o funcionamento da máquina pode ser resumido na inserção de baldes em uma esteira, transferência dos mesmos para uma segunda esteira onde o produto será dosado e envio para uma esteira de saída onde os baldes com o produto serão tampados.

A máquina pode ser dividida em quatro estações de trabalho: dispensador de baldes, esteira de baldes, dosagem e esteira de saída. A Figura 4.1 mostra uma visão real da máquina.

Figura 4.1 – Visão geral da máquina de envase

Fonte: Ilustração própria.

O dispensador de balde é o ponto inicial do processo. Esta estação consiste na retirada dos baldes de uma pilha e inserção do mesmo na esteira de entrada da máquina. A máquina possui um dispensador para envase do produto em baldes de 15 kg e dois dispensadores para baldes de 3 kg. O controle dos dispensadores é feito por elementos pneumáticos. Para segurar a pilha de baldes, são utilizados cilindros pneumáticos: seis pares em seis alturas (12 cilindros) para baldes de 15 kg e um par em cinco alturas para baldes de 3 kg. A Figura 4.2 mostra a esquerda o dispensador de baldes de 3kg em detalhe. Além do

sistema de freios da pilha de baldes, outro mecanismo realiza a retirada do balde. Esse sistema possui um par de freios que libera a retirada do balde da pilha. Um par de cilindros pneumáticos com pinças retiram o balde da pilha. Um cilindro na posição vertical executa a descida do balde que é solto na esteira após liberação de um segundo par de freios. A Figura 4.2 ilustra este sistema a direita.

Figura 4.2 – Dispensador de baldes

Fonte: Ilustração própria.

A segunda estação da máquina é a esteira de baldes, a qual recebe os baldes do dispensador e envia-os para a estação de dosagem. A esteira é movimentada por um conjunto motor/redutor, sendo o motor elétrico de 1730 rpm e 0,37 kW. O motor é controlado pelo inversor monofásico Altivar 31 da fabricante Telemecanique. Um par de baldes é enviado por vez para a estação de dosagem devido a vazão dos dosadores. Para detectar a posição dos baldes na esteira, 4 sensores fotoelétricos de reflexão difusa são utilizados. Esses sensores são colocados em uma posição que permite identificar se os baldes estão prontos para serem enviados para a estação de dosagem. Juntamente com os sensores, estão quatro cilindros empurradores com sensores magnéticos de posição que enviam os baldes para a próxima esteira. Além dos empurradores, há quatro barreiras que são levantadas por cilindros pneumáticos que encontram-se embaixo da esteira.

Figura 4.3 – Esteira de baldes mostrada em detalhes.

Fonte: Ilustração própria.

A terceira estação consiste do sistema de dosagem do produto da máquina (Figura 4.4). Como a máquina realiza sempre a dosagem em pares de baldes, a estação possui duas esteiras separadas que empurra dois baldes de cada vez para a esteira de saída. Cada esteira é movimentada por um motor de 0,15 kW e 1720 rpm. Assim como o motor da esteira de baldes, os motores das esteiras de balança são controlados por inversores Altivar 31 da Telemecanique. A estação de dosagem, conta com dosadores com acionamento pneumático. Além disso, na parte inferior da esteira, existem cilindros pneumáticos que levantam os baldes. Esses cilindros estão acoplados às células de carga, isto é, eles transferem a carga para as balanças, liberando a dosagem de acordo com o peso que está sendo medido.

Figura 4.4 – Vista detalhada da estação de dosagem.

Fonte: Ilustração própria.

A pesagem dos baldes é realizada pela célula de carga modelo PW15AHC3 com capacidade nominal de 20 kg da fabricante HBM (Figura 4.5). O princípio de funcionamento é baseado na variação de resistência de acordo com a pressão na qual a célula é submetida e uma ponte de *Wheatstone* é utilizada para geração de um sinal elétrico proporcional ao peso. Em conjunto com a célula de carga, um transdutor eletrônico é utilizado para leitura do sinal da mesma e conexão com um barramento de comunicação CANopen ou DeviceNet. O modelo do transdutor utilizado é o AED 9401A do mesmo fabricante HBM. A Figura 4.5 mostra a célula de carga utilizada com um esquema de aplicação parecido com o utilizado no processo.

Figura 4.5 – Célula de carga modelo PW15AH e esquema de aplicação.

Fonte: Datasheet da célula de carga PW15AH [8] com adaptações.

A última estação de trabalho da máquina é a esteira de saída. Esta esteira é movimentada por um conjunto motor/redutor igual ao da esteira de baldes e o controle também é feito pelo mesmo inversor. A esteira possui dois pontos de parada que são utilizados de acordo com o tamanho do balde que está sendo utilizado para envase. Os pontos de paradas ficam logo abaixo dos dispensadores de tampas. Cada posição de parada é detectada por um sensor fotoelétrico de reflexão difusa que disparam o levantamento pneumático das barreiras que freiam os baldes. Assim como as pilhas de baldes, a pilha de tampas possui freios em diversas alturas para segurar as tampas: 6 pares de cilindros pneumáticos compactos em 6 alturas para tampas de 15 kg e 4 pares e 4 alturas para tampas de 3 kg. A Figura 4.6 mostra uma visão geral da estação do dispensador de tampas trabalhando com envase do produto em baldes de 3 kg.

Figura 4.6 – Dispensador de tampas.

Fonte: Ilustração própria.

O sistema que realiza a retirada das tampas e posicionamento das mesmas nos baldes também pode ser observado na Figura 4.6. A retirada da tampa é feita por uma ventosa ligada a um sistema de vácuo. Ao prender a tampa e iniciar a descida, o braço mecânico é girado em 180 graus por um atuador pneumático giratório. Por fim, a ventosa libera a tampa ao alcançar o balde. Como a altura em que as tampas são puxadas e a altura em que as tampas são liberadas varia de acordo com o tamanho do balde, o sistema de subida e descida do braço giratório é controlado por um sistema de servoacionamento. O motor possui encoder incremental próprio e é controlado pelo servo drive *MOVIDRIVE* MDX61B. Ambos são fabricados pela SEW Eurodrive. Este servo drive, juntamente com a interface DFD11B, permite o controle do servo motor via rede DeviceNet por um CLP.

Durante a descrição do *hardware* da máquina em estudo, vários atuadores pneumáticos foram mostrados. O acionamento de todos esses componentes é controlado por um bloco de válvulas localizado na parte inferior da esteira de saída. As válvulas utilizadas possuem 5 vias de trabalho, duas posições e acionamento por solenoides de 24 Vdc. A Figura 4.7 ilustra esse sistema.

Figura 4.7 – Parte do bloco de válvulas que realiza o controle pneumático.

Fonte: Ilustração própria.

O sistema de automação pode ser controlado pelo painel de botões e a interface homem máquina encontrados no painel geral da máquina. Pelo painel de botões o operador pode ligar e desligar a máquina, selecionar o modo "teste", ligar e desligar partes especificas, como dispensador de baldes, dosador, dispensador de tampas, selecionar modo manual ou automático, selecionar qual balde será utilizado (3 kg ou 15 kg) e observar estados da máquina. Além dos comandos via botoeiras, o operador pode fazer ajustes mais avançados por meio da interface homem máquina (IHM) PanelView 300 Micro da Allen-Bradley. Esta interface está ligada ao controlador por meio de uma porta RS-232. A tela de exibição é monocromática e o comando via teclado. A Figura 4.8 mostra o painel de botoeiras da máquina e a IHM citada.

Figura 4.8 – Comandos disponíveis para o operador da máquina.

Fonte: Ilustração própria.

Por fim, o principal *hardware* utilizado: o CLP MicroLogix 1500 modelo 28BXB da Allen-Bradley. Este controlador do tipo compacto, isto é, fonte, processador, entradas e saída em um único módulo, é o mais avançado da família MicroLogix. Possui 16 entradas de 24 Vdc Sink/Source, 6 saídas a relé e 6 saídas por transistor (ambas de 24 Vdc). Oito da suas entradas e duas saídas suportam frequências de até 20 kHz. Suporta comunicação DF1 ou DH-485. Este CLP suporta cartões de expansão sem a necessidade de um rack. Foram utilizados 4 cartões de expansão de entradas/saídas: um cartão sink/source com 16 entradas, dois cartões de saída do tipo source com 16 pontos cada e um cartão com 32 saídas do também do tipo source (todos os cartões 24 Vdc). Para comunicação com as células de carga e o controlador do servo motor, um cartão de comunicação DeviceNet modelo 17690-SDN é utilizado. Este cria uma imagem de dados no controlador de 266 palavras de entrada, sendo 66 reservadas pelo scanner, e 182 palavras de saída sendo apenas 2 reservadas pelo scanner. A figura mostra o controlador utilizado e os seus cartões de expansão. Vale observa o número de cada slot para melhor compreensão do endereçamento: o controlador ocupa o slot 0, o scanner o slot 1 e assim por diante.

Figura 4.9 – Controlador do caso referência.

Fonte: Ilustração própria.

4.2 Conclusões

A apresentação dos componentes de *hardware* utilizados em um sistema de automação real é importante para a familiarização com os elementos sensores e atuadores comumente utilizados em soluções de automação do mercado atual. Além disso, a abordagem dos componentes da máquina envasadora estudada é a base para a compreensão e simulação da lógica implementada via programação em *ladder* que será discutida nos capítulos seguintes. Para a definição das funções e parâmetros controlados pela a IHM desenvolvida neste trabalho, a apresentação do *hardware* utilizado no caso em estudo é de grande contribuição.

5 LÓGICA DO CASO ESTUDADO

Conhecido os componentes de *hardware*, a teoria de programação em ladder e os softwares necessários para implementação de um projeto de automação, é possível analisar e reproduzir as soluções de programação utilizadas no desenvolvimento do caso referência.

O programa em funcionamento no caso em estudo possui 25 arquivos de programas ladder. Por isso, a análise de cada programa implementado tornaria o trabalho muito extenso e repetitivo, pois, como já foi visto no capítulo anterior, a máquina pode rodar com dois tipos de balde, resultando em sub-rotinas duplicadas com parâmetros diferentes, mas com o mesmo princípio de funcionamento. Assim, optou-se pela análise da lógica implementada em rotinas que visam o funcionamento da máquina com baldes de 15 kg.

5.1 Configuração do CLP

Um dos primeiros procedimentos a ser executado durante o desenvolvimento de um projeto de automação é a configuração do CLP, pois é preciso testar o seu funcionamento e reconhecer os cartões conectados para ficar ciente de quais endereços serão utilizados. O CLP utilizado, MicroLogix 1500, vem configurado de fábrica para utilizar o protocolo de comunicação DF1 *full-duplex*. Utilizando um cabo adaptador é possível conectar o CLP a um computador pessoal e configurar o respectivo driver no *software RSLinx* por meio da opção *“Auto-Configure”*. Vale ressaltar que os parâmetros configurados de fábrica para a rede DF1 do MicroLogix são os mesmos que os parâmetros pré-configurados na criação do driver pelo *RSLinx*.

Estabelecida a conexão com o CLP, um novo projeto deve ser criado no *software* RSLogix 500 e o modelo exato do controlador deve ser informado. Com o driver de comunicação configurado, a configuração dos cartões de expansão conectados ao controlador se torna mais fácil, pois, nessa situação, o *software* de programação permite a leitura automática dos cartões utilizados. A figura mostra os slots e cartões configurados no CLP utilizado no caso em estudo. A configuração dos parâmetros de comunicação também é facilitada pelo o uso da opção “*Who Active”* que selecionar um dispositivo reconhecido pelo computador com controlador a ser utilizado no projeto.

Figura 5.1 – Slots e cartões configurados.

#	Part #	Description
0	Bul.1764	Micrologix 1500 LRP Series C
1	1769-SDN	DeviceNetScanner
2	1769-IQ16	16-Input 10/30 VDC
3	1769-OB32	32-Output High Density 24 VDC
4	1769-OB16P	16-Output 24 VDC Source w/ Protectio
5	1769-OB16	16-Output 24 VDC Source

Fonte: Ilustração própria obtida com o auxílio do *software* RSLogix 500.

Com o CLP conectado e devidamente configurado, será possível testar os elementos atuadores e sensores para evitar complicação na hora de depuração de erros com a lógica completamente implantada.

5.2 Configuração da rede DeviceNet

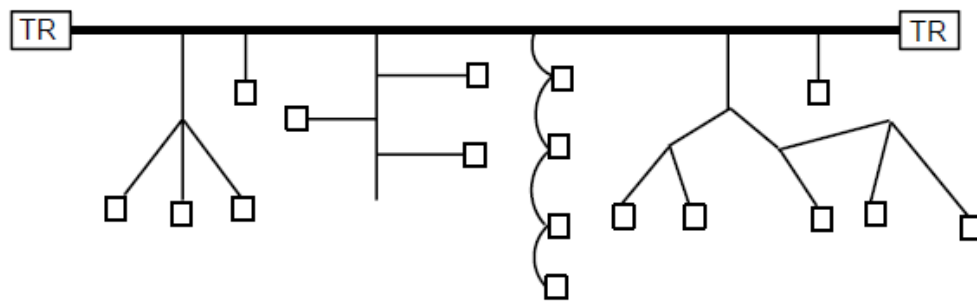
5.2.1 Características da rede DeviceNet

A rede DeviceNet é uma rede aberta direcionada a dispositivos de baixo-nível baseada no protocolo de comunicação CAN (*controller área network*) desenvolvido pela Bosch. Os padrões utilizados na rede DeviceNet são estabelecidos pela *Open DeviceNet Vendor AssociationI* (ODVA).

A rede DeviceNet suporta até 64 nós (dispositivos associados a rede) e, dependendo do tipo de ligação os componentes podem ser retirados da rede sem interromper o funcionamento dos demais equipamentos. A topologia utilizada nesta rede é do tipo barramento com derivações. Cada derivação também pode conter ramificações.

O cabo utilizado pela na rede DeviceNet transporta alimentação e sinais de comunicação junto. A cabo é composto por 5 fios: dois de alimentação, dois de comunicação (CAN_L e CAN_H) e um *shield*. Os cabos são padronizados em quatro tipos: grosso, médio, fino e chato. A escolha do cabo interfere na taxa máxima de transmissão. Geralmente, o cabo grosso é utilizado no barramento e o cabo fino nas derivações.

As conexões dos dispositivos podem utilizar de linhas de derivação que devem ser conectadas ao barramento por meio de *taps* (um *tap* pode distribuir mais de uma derivação) ou seguindo o estilo aberto onde o barramento é partido para conexão de um dispositivo. Este último estilo pode ser visto na quarta derivação da topologia mostrada na Figura 5.2. O fim do barramento deve indicado pela conexão de uma resistência de 120 ohms chamada de *Terminating resistor* (TR).

Figura 5.2 – Topologia genérica de uma rede DeviceNet.

Fonte: Ilustração própria obtida com o auxílio do *software* RSLogix 500.

O parâmetro principal a ser configurado em uma rede DeviceNet é a sua taxa de transmissão ou *baud rate*. A velocidade máxima para aplicação desenvolvida pode ser definida por três critérios: tipo de cabo utilizada, distância máxima entre dois dispositivos e valor acumulado da distância das derivações. Os valores disponíveis para configuração da taxa de transmissão da rede são: 125 kbps, 250 kbps e 500 kbps. A tabela mostra a velocidade máxima de acordo com os parâmetros citados.

Tabela 5.1 – Seleção de velocidade da rede DeviceNet.

Velocidade	**Distância máxima (m)**				**Distância acumulada (m)**
	Cabo chato	Cabo grosso	Cabo médio	Cabo fino	
125 kbps	420	500	300	100	156
250 kbps	200	250	250	100	78
500 kbps	75	100	100	100	39

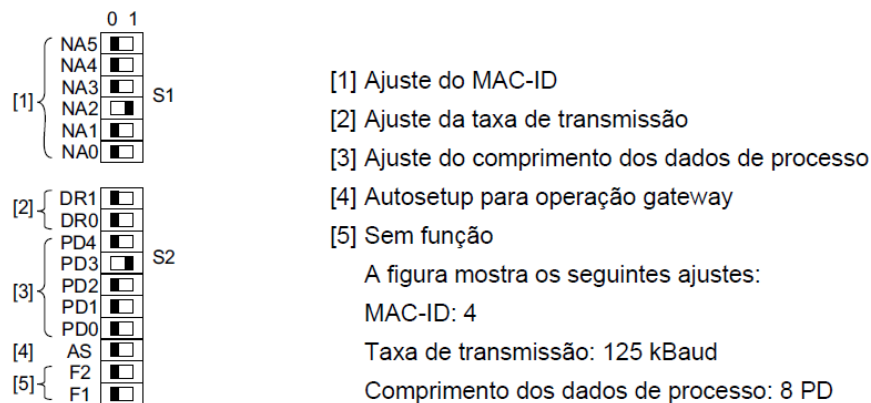
Fonte: Tabela própria com dados de [10]

5.2.2 Configuração do caso em estudo

Após o mapeamento dos cartões de entradas e saídas comuns utilizados, os outros endereços a serem mapeados são os da rede DeviceNet. Primeiramente, com apenas o computador conectado à rede, o driver da rede deve ser configurado via *RSLinx*. Com a interface 1770-KFD da Allen-Bradley é possível ligar o computador à rede e, após configurado o driver de comunicação no *RSLinx*, mudar a velocidade de transmissão da rede para 500 kbps. Essa velocidade foi escolhida porque o barramento tem um curto comprimento e não possui derivações, possibilitando a escolha da velocidade máxima da rede.

Em seguida, com o *software* RSNetWorx, é possível modificar o endereço do cartão scanner para confirmar que o scanner não estará em um endereço duplicado. Após isso, deve ser realizado o comissionamento do primeiro dispositivo conectado à rede: o conversor

MOVIDRIVE MDX61B. Este servo drive possui uma placa (DFD11B) que possibilita a conexão do conversor em uma rede DeviceNet. Nesta placa, chaves seletoras possibilitam selecionar a taxa de transmissão, o endereço e a quantidades de dados trocados nas mensagens. A velocidade é selecionada com os seguintes valores de DR1 e DR0: 0 seleciona 125 kbps, 1 seleciona 250 kbps, 2 seleciona 500 kbps e 3 não é válido [12]. A figura mostra o que parâmetro cada grupo de chave representa e um exemplo de configuração da placa DFD11B.

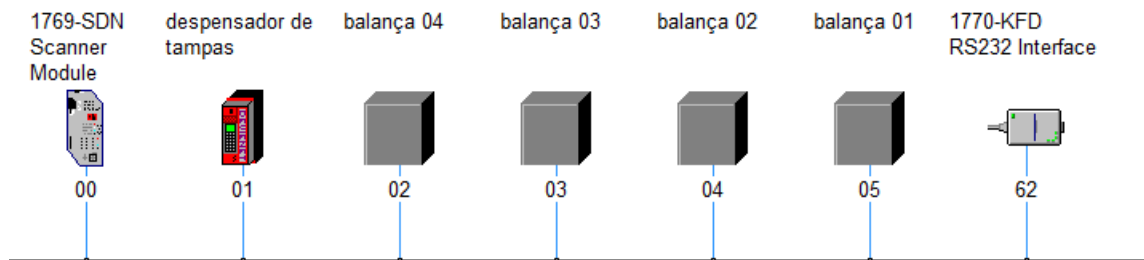
Figura 5.3 – Comissionamento do MOVIDRIVE MDX61B.

Fonte: Manual da interface DFD11B (2007) [12].

No caso em estudo, a interface do servodrive foi configurada com endereço 1, taxa de transmissão 500 kbps, 3 palavras de entrada e 3 palavras se saída. Após o comissionamento do dispositivo, é possível configurar o restante dos parâmetros diretamente no *software* RSNetWorx.

Os transdutores das células de carga possuem uma chave que permite a desconexão do dispositivo sem interferir no restante da rede, o que é uma ótima opção já que o barramento da rede está conectado pelo estilo aberto em cada célula de carga. Além desse *switch*, outro é utilizado para comutar o tipo de rede que o transdutor está conectado entre CAN e DeviceNet. Também é possível habilitar a resistência terminal se o transdutor for o ultimo dispositivo da rede. O nó padrão de equipamentos não configurados é o 63. Por isso, esse endereço deve permanecer livre em uma rede DeviceNet para permitir a fácil configuração de novos equipamentos. Aproveitando-se da chave de remoção do transdutor da rede, cada célula de carga de ser configura separadamente via RSNetWorx.

Por fim, via *software* RSNetWorx, o scanner deve ser especificado para o slot 1 e plataforma MicroLogix (palavras de 16 bits). Os dispositivos devem ser habilitados na aba *scanlist* e nas abas *input* e *output* a opção *automap* deve ser utilizada, resultando na

configuração final de 12 bytes de entrada do servo drive, 13 bytes de entrada para cada balança, 12 bytes de saída do servo drive e 2 bytes de saída para cada balança (todas com tipo de comunicação *polled*). A figura mostra a configuração final da rede com o endereço de cada dispositivo.

Figura 5.4 – Rede do caso em estudo configurada via RSNetWorx.

Fonte: Ilustração própria obtida com auxílio do *software* RSNetWorx.

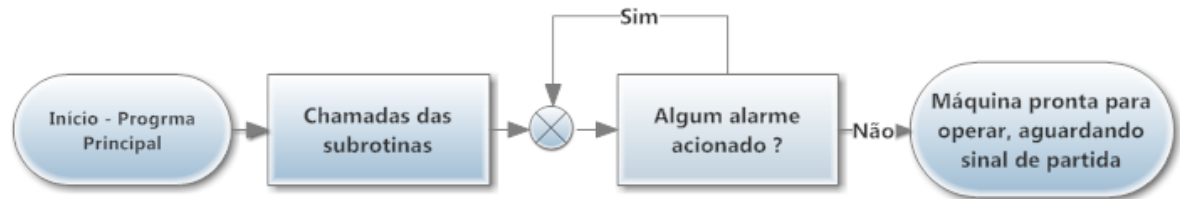
5.3 Programa ladder do caso em estudo

Como já foi mencionado, alguns arquivos de programas ladder não serão analisados devido a semelhança com outros arquivos do projeto. Por exemplo, o código desenvolvido para o controle do dispensador de baldes de 15 kg é muito semelhante ao código do dispensador de baldes de 3 kg. É importante ressaltar que a sequência seguida pelos tópicos terciários deste capítulo é a mesma sequência de scan dos arquivos ladder do projeto estudado.

5.3.1 Programa principal

Como foi visto no Capítulo 3, a máquina apresentada no caso em estudo pode facilmente ser dividida em estações. O mesmo princípio é utilizado no desenvolvimento do arquivo ladder principal, isto é, o código do projeto foi dividido em diversas sub-rotinas para otimizar a execução. Neste primeiro arquivo ladder, é bom ressaltar a importância da utilização de símbolos para os dados utilizados no projeto. A utilização de símbolos consiste em nomear as variáveis para que um endereço possa ser referenciado pelo nome escolhido. A definição de símbolos facilita bastante o desenvolvimento do projeto. No projeto do caso em estudo, tanto as variáveis de entrada e saída físicas como os parâmetros internos estão devidamente referenciados. Uma tabela com a descrição das entradas em saída é apresentada no Anexo A. Além da chamada de todas as sub-rotinas do projeto, o arquivo ladder número 2 define a variável interna (B3:1/11) que determina se a máquina está em um estado pronto para operação. A máquina é defina como pronta para funcionar quando não houver a ocorrência de nenhum alarme, o servo motor estiver referenciado e houver a ausência do comando de *Clean*

in Place (CIP). CIP é um termo bastante utilizando na indústria alimentícia que consiste na limpeza da máquina durante a troca do produto a ser envasado. Estando a máquina no estado pronto para operar, a máquina pode ser ligada pela botoeira de partida (intertravada com a botoeira de parada). O fluxograma ajuda a esclarecer a execução do programa ladder principal.

Figura 5.5 – Fluxograma programa ladder principal

Fonte: Ilustração própria.

5.3.2 LAD 3 – Freio da pilha de baldes

Neste arquivo, é realizado o controle dos freios pilha de baldes. O freio da pilha de baldes devem trabalhar em conjunto com o sistema que puxa os baldes da pilha. Por isso, o início do ciclo depende da variável *cames_00* vinda do processo do dispensador de baldes. Primeiramente, para iniciar os ciclos do dispensador de baldes, o sistema dispensador de baldes de estar ligado e em modo manual (ambas entradas definas pelo painel de operação). Estando estas duas condições verdadeiras, a variável CAMES_00_15K, que indica o início de cada ciclo do dispensador de baldes, gera um pulso de subida, por meio da função ONS, que incrementa o número de ciclos do sistema de freios por meio de uma instrução de adição. Quando o número de ciclos for maior que 2, o sistema de freios será acionado e o número de ciclos zerado. Três temporizadores em cascata contam 1 segundo cada. Durante o primeiro segundo, o freio inferior deve ser liberado. No 2º segundo, o freio superior é liberado e, após o terceiro segundo, variável que libera o acionamento do ciclo de freios é resetada.

5.3.3 LAD 5 – Dispensador de balde

Para iniciar o dispensador baldes, diversas condições devem ser satisfeitas como verificar se a máquina está ligada, se o comando para ligar o dispensador via painel foi acionado e se a esteira de baldes está ligada, pois esta esteira é desligada sempre que há 4 baldes prontos para serem enviados para a dosagem e não deve ser alimentada enquanto estiver sobrecarregada.

A dispensa de baldes na esteira é um processo cíclico e sem variações. Por isso, uma solução antiga foi utilizada para automação desta tarefa: virtualização de um temporizador cíclico eletromecânico. Estes temporizadores são utilizados em processos com sequência e tempo definidos. Eles são constituídos de um motor e um eixo com cames. As cames são projetadas de tal forma que, em determinado ponto do giro do temporizador, cada uma vai acionar um interruptor. A virtualização deste temporizador consiste em incrementar um ângulo virtual por um temporizador. As diversas tarefas do processo cíclico são acionadas de acordo com o intervalo do ângulo virtual em que elas estão programadas. No caso do programa ladder do dispensador de baldes, um temporizador realiza um incremento de 2,7 graus a cada 0,05 segundos. Em seguida, é verificado se o ângulo ultrapassou o valor máximo de 360, zerando o ângulo por meio de uma instrução SUB. O valor do ângulo é arredondado por meio da movimentação de um valor float para um do tipo inteiro. Em seguida, uma instrução LIM defini os limites do intervalo do ângulo virtual para cada came. No final os atuadores do dispensador de baldes são acionados de acordo com as cames ativos. É importante ressaltar que o intervalo de 0 a 10 graus é ponto de repouso do sistema e a máquina só para estando o dispensador no came 0, isto é, no caso de um alarme ser acionado, o dispensador irá completar o ciclo e, só então, parar.

Tabela 5.2 – Parâmetros do dispensador de baldes.

Regulagem dispensador de baldes

Processo	Comando	Ângulo virtual
Parada	Liga	0
	Desliga	10
Prendedor de balde	Liga	200
	Desliga	360
Dispensador de balde	Liga	40
	Desliga	60
Separador de balde	Liga	70
	Desliga	250
Puxador de balde	Liga	65
	Desliga	130

Fonte: Tabela própria.

5.3.4 LAD 8 - Contagem de produção

O arquivo de programa 8 conta o número de baldes produzidos na linha. O valor é informado via IHM e pode ser resetado sempre que uma nova jornada de produção iniciar-se.

A contagem é feita por pulsos de transição de subida sempre que a dosagem de um par de baldes, balanças 1 e 2 ou balanças 3 e 4, for concluída. A contagem é zerada se o valor da produção ultrapassar o armazenamento de uma variável do tipo long.

5.3.5 LAD 9 – Pesagem

O nono arquivo de programa ladder, é provavelmente um dos mais complexos do projeto. Neste arquivo, é controlado o peso dos baldes e, consequentemente, a abertura dos dosadores. Diferentemente de outras partes da máquina que são divididas de acordo com o tamanho do balde, este ladder realiza a configuração do sistema de pesagem para qualquer tamanho de balde que esteja rodando na máquina.

A primeira ação tomada no ladder de pesagem é a garantia de que os dosadores estarão fechados durante o comando de CIP da máquina. Também no início do programa, os cilindros que levantam as balanças devem retornar para suas posições de retorno, permitindo a entrada de baldes na estação de dosagem e impedindo a transferência de carga para a balança fora do processo de dosagem. Em seguida de acordo com a chave de seleção de peso dos baldes, é realizada a configuração do valor do peso a ser medido em cada balança, de um valor para antecipação da abertura dos dosadores e o tempo que a esteira da balança de dosagem deve permanecer ligada para transferir os baldes para a esteira de saída. A escala do valor medido pelas células de carga é configurada por meio do *software* disponibilizado gratuitamente pela fabricante da célula: AED Panel 32. Neste programa, a configuração de diversos parâmetros é simplificada, permitindo até o comissionamento do dispositivo na rede DeviceNet. A balança permite um complexo sistema de dosagem automática, que verifica tolerâncias e manipula a válvula dosadora. Porém, devido a configuração mais complexa desse sistema e a maior exigência de harware, optou-se por uma solução por comando do CLP com boa precisão e configuração mais simples. A Tabela 5.3 mostra os parâmetros da célula de carga que podem ser modificados.

Tabela 5.3 – Parâmetros da célula de carga via DeviceNet.

Dados de entrada		
Byte	**Parte da palavra**	**Descrição**
0		IMD value, 2
1	Menos significativa	MSV value (measured value)
2		MSV value (measured value)
3		MSV value (measured value)
4	Mais significativa	MSV value (measured value)
5	Menos significativa	MSV status (measurement

		status)
6	Mais significativa	MSV status (measurement status)
7	Menos significativa	FRS value (dosing result)
8		FRS value (dosing result)
9		FRS value (dosing result)
10	Mais significativa	FRS value (dosing result)
11	Menos significativa	FRS status (dosing status)
12	Mais significativa	FRS status (dosing status)
Dados de Saída		
Byte	**Bit**	**Descrição**
0	0	TAR – Tare
	1	TAS – Gross/net selection
	2	CSN – Clear dosing result
	3	RUN – Start dosing process
	4	BRK – End dosing process
	5	CTR – Clear trigger result
	6	CDL – Zeroing
	7	CPV – Clear peak value
1	0	Reserved
	1	Reserved
	2	Output 1 desired state
	3	Output 2 desired state
	4	Output 3 desired state
	5	Output 4 desired state
	6	Output 5 desired state
	7	Output 6 desired state

Fonte: Tabela própria com dados de [13]

Após a configuração dos pesos objetivos para cada tipo de balde, o programa verifica se há baldes na esteira do dosador por meio de uma variável de estado definida no programa dos empurradores (visto na seção 5.3.8). Sendo esta condição verdadeira, o estado da máquina é definido como: “Em processo de dosagem”. Nesta, situação, os levantadores da balança são acionados e é enviado um comando para tarar as duas balanças que contém carga. Após a tara das balanças, os valores medidos possuem um offset que só será modificado com a entrada dos dois próximos baldes. Após estes passos, as duas válvulas dosadoras com baldes serão liberadas. O acionamento de um par de válvulas está intertravado com o estado atual do outro par. Porém, um peso de antecipação é configurado apenas para o balde de 15 kg que permite o início da dosagem no outro par de válvulas quando a dosagem do mesmo está

próxima do fim. As válvulas dosadoras continuam liberadas até que o peso lido na rede DeviceNet supere o peso estabelecido para cada balde (instrução “maior que” utilizada).

Com a dosagem concluída dentro da tolerância, a esteira da balança é acionada por um tempo suficiente para enviar as balanças para a esteira de saída. Em seguida, todos os bits virtuais de estado da dosagem são resetados. O Apêndice A contém um fluxograma que esclarece a execução do código ladder de dosagem.

5.3.6 LAD 13 – Dispensador de tampas de 15 kg

O programa do dispensador de tampas é semelhante ao utilizado para o dispensador de baldes, utilizando também o sistema de cames. Para iniciar o processo do dispensador de tampas para baldes de 15 kg, é preciso confirmar, por meio de um sensor, que há um balde na estação de tampas e o balde anterior já saiu da estação de tampas. Sendo a condição favorável, a variável que liga o dispensador de tampas é acionada. Em seguida, o sensor de tampas do balde pequeno verifica se está passando um balde pela estação. O dispensador só será liberado, quando houver a confirmação que o balde em passagem já saiu da estação (temporizador confirma sinal do sensor da estação de balde pequeno desligado). Liberado o ínicio do ciclo do dispensador, um sistema de cames semelhante ao do dispensador de baldes é acionado. Um temporizador é utilizado como clock de 50 ms para o incremento do ângulo virtual. O incremento utilizado é de 2,7 graus. Em seguida, é verificado se o ângulo virtual ultrapassou 360 graus, resultando em um reset em caso de verdade.

O dispensador possui cinco processos cíclicos acionados por cames. Em um intervalo de 0 a 20 graus, o dispensador se encontra em estado de repouso. De 50 até 190 graus o vácuo que prende a tampa no braço giratório é acionado. Iniciando junto com o vácuo da tampa, o dispensador de tampas (freio da pilha) permanece acionado durante 20 graus do ciclo. O atuador pneumático giratório é acionado por a partir de 95 graus e desligado com 240 graus. O 4º came é utilizado para acionamento do servo motor que será discutido mais adiante. O quinto came também é utilizado em outro código que será discutido.

Assim que o came 1 é concluído, a barreira que mantém o balde na posição é desligada e espera a passagem do balde. Em seguida, é religada para posicionar o próximo balde. A Tabela 5.4 resume o ciclo do dispensador de baldes.

Tabela 5.4 – Parâmetros dispensador de tampas.

Regulagem dispensador de tampas		
Processo	Comando	Ângulo virtual
Parada	Liga	0
	Desliga	20
Vácuo da tampa	Liga	50
	Desliga	190
Dispensador de tampas	Liga	50
	Desliga	70
Gira tampas	Liga	95
	Desliga	240
Servo motor	Liga	80
	Desliga	240
Teste vácuo	Liga	110
	Desliga	140

Fonte: Tabela própria.

5.3.7 LAD 16 – Servo Tampa

O arquivo de 16 do projeto ladder realiza a comunicação e controle do servoacionamento de acordo com os seus respectivos cames definidos no programa dos dispensadores. Para entender o funcionamento do código, é necessário conhecer as palavras de comunicação do servodrive e os seus modos de acionamento. Como já foi visto, na configuração do servodrive para a rede DeviceNet, são utilizadas três palavras de entrada e três palavras de saída. O conteúdo dessas palavras pode ser configurado via RSNetWorx. No caso em estudo, as entradas são uma palavra de status, velocidade atual e posição atual. As saídas são: palavra de controle, posição desejada e velocidade desejada. A Figura 5.6 mostra os parâmetros das palavras de controle do drive. Para liberar o controle da posição e velocidade via DeviceNet, a entrada digita física 1 deve ser energizada por com 24 Vdc. Além disso, os parâmetros *Control source* e *Setpoint source* devem ser ajustados para *FIELDBUS* por meio do *software* de configuração da rede DeviceNet. A fabricante SEW Eurodrive disponibiliza um *software* com assistente de configuração para *startup* do drive. Tempo de rampa, aplicação desejada, cálculo de escalas, modo de referência e outros podem ser facilmente ajustados pelo programa.

Figura 5.6 – Palavras de controle do servodrive MDX61B

OUTPUT1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

/SWLS/rampa1/2

PosMode 2^0

Seleção de modo 2^1

Seleção de modo 2^0

Pos. pulsos / PosContínuo

Pulsos+

Início

Regulador bloqueado/ liberação

Liberação/ parada rápida

Liberação/parada

Controle de retenção

Comutação de rampas

Comutação de con- junto de parâmetros

Reset irregularidade

Reservado

INPUT1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bit 8 .. bit 15: Estado do con- versor / Cód. irregul.

Irregularidade sensor de material

Conversor pronto

Referência IPOS (= Acionamento referenciado)

Posição de destino alcançada

Freio liberado

Irregularidade / aviso

Chave fim de curso horária ativa

Chave fim de curso antihorária ativa

Fonte: Manual de aplicação do MOVIDRIVE MDX61B (2005) [14].

O modo do funcionamento do servodrive é controlado pelos bits 11 e 12 da palavra de saída. Os valores um, dois e três selecionam modo JOG (Posicionamento por pulsos), modo de referenciamento e modo automático respectivamente. Os bits 13 e 14 selecionam o modo de posicionamento. Os valores zero, um, dois e três selecionam posicionamento absoluto, relativo, distância restante horária e distância restante anti-horária

respectivamente. O modo absoluto posiciona o servo de acordo com a posição referência e a posição destino. No modo relativo, a posição referência passa a ser a posição atual. Os modos de distância restante dependem de gatilhos de posição do material para definirem o posicionamento e não foram utilizados. Para referenciar a posição do drive, um sensor de referência é utilizado, enviando um sinal alto quando o drive passa pela posição “zero”. O esquema das ligações necessárias para o funcionamento do servodrive com controle via rede está no Anexo B.

O programa é iniciado com a leitura da posição do servo motor. Em seguida, os cursos de subida e descida, velocidades de subida e descida e posição em caso de falha são definidos para cada tipo de balde. Feita a configuração dos parâmetros a serem enviado para o servodrive, o programa analisa se há falha no conversor. Em caso de falha, os comandos de inibir e ligar o servodrive são zerados (bits 0 a 3 da palavra de controle). Não havendo falha, os bits de liberação do controlador são setados. Com o bit 1 da palavra de controle de entrada indicando controlador pronto, o modo de funcionamento é alterado para referenciamento e o comando de início é enviado, deixando o servo motor referenciado e pronto para o processo. A máquina de envase possui uma chave de seleção de modo teste ou automático, permitindo, em modo teste, que o servodrive seja utilizado em modo JOG para realização de ajustes. Controle do modo JOG é feito via IHM existente. Estando o servo motor referenciado e modo JOG desativado, o modo de posicionamento do servodrive é alterado para automático com posicionamento relativo. Os valores registradores contendo as referências de posição e velocidade são enviados para o controlador e o posicionamento do servo motor é iniciado pelo bit 8 da palavra de controle (início). No primeiro posicionamento, o servo posiciona o braço giratório da estação de tampas logo a baixo da tampa para o vácuo ser ativado e a tampa puxada. Quando o came 4 do dispensador de tampas for acionado, o valor de posição destino é trocado pelo valor de descida já armazenado na configuração dos baldes. No mesmo instante, o servo desce o braço giratório para a posição adequada. Quando o came 4 é desabilitado, a posição inicial do servo motor assume novamente o valor de posição destino. Em caso de falha no vácuo das ventosas das tampas, o servo é movimento para uma posição de falha. A Tabela 5.5 mostra os parâmetros utilizados para configuração do servo.

Tabela 5.5 – Parâmetros servo motor.

Regulagem servo motor	
Parâmetro	Valor
Velocidade de descida (rpm)	1100
Velocidade de subida (rpm)	1500
Rampa de aceleração	70
Rampa de desaceleração	100
Curso de subida	15
Curso de descida	1400
Curso de falha	400

Fonte: Tabela própria.

5.3.8 LAD 18 – Empurradores de baldes

O código ladder do empurrador de baldes é responsável pelo controle da esteira de entrada, dos paradores dos baldes e do envio dos baldes para a estação de dosagem. Vale lembrar que a esteira de entrada conta com quatro sensores (um sensor para cada dosador), quadro barreiras e quatro pistões empurradores.

A esteira é desligada sempre que houver quatro baldes prontos para serem enviados para a estação de dosagem. Caso falte qualquer balde, a esteira estará ligada. Em seguida, o código transfere valores para o presets dos temporizadores que acionam as barreiras de acordo com a chave de seleção do peso dos baldes. As barreiras possuem um temporizador como filtro, isto é, quando o sensor de determinada barreira está ativo, o levantamento da barreira é adiado para que o balde possa passar. Em seguida, a barreira é acionada e um temporizador aguarda a chegada do balde no próximo empurrador. Se algum empurrador ficar sem balde a montante da barreira, a mesma recebe um comando OTU. O sensor de posição dos empurradores a montante são utilizados para impedir a passagem de baldes com os próximos empurradores acionados.

Os empurradores de baldes são acionados em duplas, isto é, são acionados empurradores um e dois ou três e quatro. Para acionar cada dupla de empurradores, diversas condições devem ser satisfeitas: os dois empurradores devem possuir baldes, suas barreiras devem estar acionadas, a auxiliar indicativa da presença de baldes nas balanças deve ser falsa, a esteira das balanças que vão receber os baldes devem estar desligadas e a estação de dosagem deve estar ligada. Após serem acionados, a presença de baldes nas balanças é indicada por um bit virtual e um temporizador de 1,6 segundo desativa os empurradores. No

final do código, algumas condições são impostas para desligar o dispensador de baldes, como presença de balde no empurrador mais próximo do dispensador.

5.3.9 Demais códigos

Diversos arquivos do projeto se utilizam dos alarmes para garantirem segurança nas suas operações. O projeto conta com uma rotina separada para definição dos alarmes. Nesta rotina, cada arquivo conta com uma variável virtual que é setada em uma condição de falha. Por exemplo, o alarme indicativo de falta de vácuo na estação de tampas é estabelecido durante o came 5 do dispensador de tampas por meio de uma vácuostato, havendo quebra do vácuo o alarme é setado. Ao todo, são 6 alarmes definidos que ativam a sinaleira de alarme e desabilitam determinados processos. O décimo nono arquivo (BTD_CÉLULAS) é bastante simples e repetitivo. Neste arquivo, os dados da célula de carga são copiados para endereços que são utilizados no código de pesagem. Como alguns dados da balança possuem 32 bits, a cópia da parte mais significativa da palavra é feita bit a bit para um registrador do tipo long que já contém a parte menos significativa.

Há uma rotina para ajustar o relógio do CLP via IHM que copia valores informados via IHM para o arquivo RTC do controlador que contém data e hora. Para copiar os valores, foi utilizado o comando copy que, diferentemente da instrução MOV, transfere o padrão de bits da palavra sem realizar conversão de formatos. Uma rotina denominada mensagens é utilizada para definir o número da tela a ser exibida. Nesta rotina, a tela da IHM é definida de acordo com a ocorrência de algum alarme ou modo de produção. A rotina de primeiro ciclo de scan, utilizada no programa principal, realizada o reset de variáveis auxiliares de processo e habilita o cartão scanner da rede DeviceNet. Por fim, uma rotina denominada CIP é executada quando a máquina está em modo de limpeza. O arquivo conta com dez temporizadores em cascata que definem o momento de energização das auxiliares de controle das válvulas dosadoras. As auxiliares de CIP das dosadoras, são utilizados no ladder de pesagem para abertura das válvulas. Neste arquivo, também é realizado o controle da válvula geral alimentadora dos dosadores. Em funcionamento normal, ela é controlada por uma saída não retentiva que é acionada sempre que uma dosadora é ativada. No modo CIP, ela permanece aberta até o sétimo temporizador terminar a contagem.

5.4 Conclusões

O quinto capítulo do desenvolvimento apresenta as soluções de programação em ladder utilizadas no caso em estudo. Neste capitulo, uma abordagem mais detalhada sobre a

rede utilizada no caso em estudo é feita. A rede DeviceNet, quando comparada as demais redes industriais, é uma das que impõe mais dificuldades ao usuário. Para esta rede, cada equipamento deve ser comissionado e uma configuração de rede tem que ser desenvolvida. Qualquer acréscimo ou mudança na rede, resulta em uma nova configuração manual. Apesar das dificuldades impostas pela rede, o estudo deste protocolo facilita o entendimento das demais redes industriais, que geralmente possuem configurações mais simplificadas, e também facilita a compreensão sobre a manipulação dos parâmetros dos dispositivos que se comunicam com o CLP via rede. Além da abordagem sobre a rede utilizada, é apresentado o procedimento para configuração dos equipamentos que se comunicam com o CLP via DeviceNet, esclarecendo os procedimentos seguidos no planejamento da rede. Nesta etapa, ficou claro como a escolha de um caso prático baseado em equipamentos Allen-Bradley e softwares Rockwell foi acertada, pois, até em manuais de outros fabricantes, os equipamentos e softwares dessa parceria são utilizados como demonstração. Isso comprova a popularidade destas desenvolvedoras, indicando que o início do estudo de automação com ferramentas desta parceria é uma boa escolha. Por fim, o estudo do caso referência mostrou que a programação em ladder é simples, mas bastante repetitiva. A utilização da linguagem gráfica apresenta vantagens como observação do funcionamento e fácil solução de problemas, mas requerem soluções volumosas para algumas tarefas simples. Também é importante destacar que apesar da fácil programação, os processos automatizados nesta linguagem, geralmente, apresentam muitas variáveis dependentes/auxiliares o que é exige mais atenção do programador.

6 DESENVOLVIMENTO DE UMA INTERFACE HOMEM-MÁQUINA PARA O CASO EM ESTUDO

No desenvolvimento deste trabalho, já foram abordados diversos conceitos necessários para a implementação de um projeto de automação industrial, entre eles estão as noções básicas do *hardware* de um CLP, programas necessários para desenvolvimento de uma aplicação, conhecimento de sensores, atuadores e outros dispositivos necessários para a automação de uma máquina industrial (por meio do estudo de um caso real), e instruções e configurações de um programa desenvolvido com a linguagem *ladder*. Por fim, seguindo com a proposta deste trabalho de apresentar uma abordagem geral sobre as diversos conceitos e áreas da automação industrial, um último tópico será abordado: interface homem-máquina (IHM). As IHMs consistem no meio em que o usuário controlará e supervisionará a aplicação de automação. A interface pode ser implementada por um conjunto de botões, chaves seletoras e sinaleiras ou por um complexo sistema de supervisão, controle e aquisição de dados (SCADA). O tipo de interface deve ser escolhido de acordo com o grau de complexidade da automação aplicada, o grau de controle permitido ao usuário final, a distância entre o processo automatizado e o local de operação entre outros fatores. A IHM, sem dúvidas, é um elemento fundamental em um sistema automatizado, pois permite controle total sobre a aplicação, garantindo a praticidade, liberdade e eficiência do sistema implementado. Além desses fatores, a IHM é um fator decisivo na aceitação de um projeto de automação, pois a possibilidade da aproximação do usuário final com a programação, controle e configuração do sistema automatizado é um atrativo e garantia de eficiência da aplicação para o cliente.

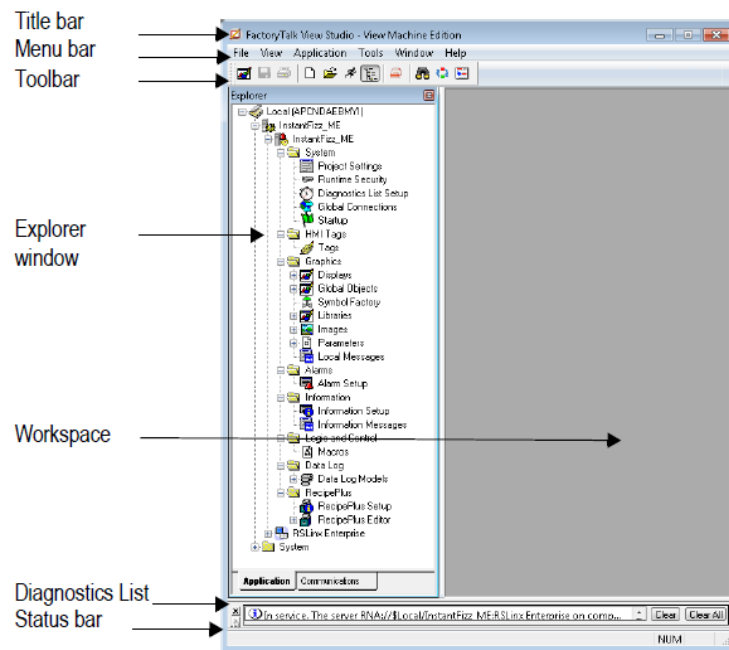
Neste capítulo, será detalhado o desenvolvimento e simulação de uma IHM que possibilita a configuração e operação da máquina do caso em estudo.

6.1 Apresentação do Factorytalk View

Os programas utilizados no decorrer deste trabalho já foram abordados no capítulo 2.4 dessa tese. Porém, optou-se por uma abordagem separada do *software Factorytalk View* devido à maior utilização deste *software* no desenvolvimento deste trabalho.

O *Factorytalk View* é a ferramenta computacional da *Rockwell* utilizada para desenvolvimento e teste de IHMs. Este *software* possui dois ambientes: *Site Edition* (SE) e *Machine Edition* (ME). Como o nome já sugere, a versão SE é utilizada para desenvolvimento de interfaces mais complexas (SCADA), com a possibilidade de criar servidores e clientes. Já

o ambiente ME é voltado para o desenvolvimento de interfaces locais para máquinas industriais, ou seja, à nível de chão de fábrica. A IHM desenvolvida neste trabalho foi criada a partir do ambiente *Machine Edition*. A Figura 6.1 mostra o ambiente de trabalho do *Factorytalk View* ME.

Figura 6.1 – Visão geral do *software Factorytalk View ME*.

Fonte: Página 2-4 do manual do *software* Factorytalk View [15].

Após o reconhecimento de diversos softwares da *Rockwell Automation*, a adaptação a um novo programa da mesma desenvolvedora é facilitada devido ao fato dos softwares apresentarem um *layout* similar uns aos outros. Como está apontado na figura, o ambiente de desenvolvimento deste *software* apresenta uma barra de menu no topo da tela que dá acesso completo às ferramentas do programa. Logo abaixo, uma barra de ferramentas possibilita acesso direto aos recursos mais utilizáveis. Essa barra é configurável e permite a adição de elementos de acordo com a necessidade do programador, pois é possível configurar acesso direto aos elementos gráficos, ferramentas de posicionamento e diversos outros grupos de objetos.

A janela de exploração permite lista e possibilita acesso aos diversos elementos da IHM desenvolvida, ou seja, este ambiente é um gerenciador dos objetos da interface. Nela, é possível acessar as configurações do sistema, *tags* da interface, telas criadas, biblietoca de imagens, configurações de alarmes, macros e diversos recursos que serão abordados do decorrer deste capítulo. Ao lado da janela de exploração, está ou *workspace* ou ambiente de

trabalho. É nesta seção do *software* que qualquer tipo de configuração é feita. Na parte inferior do *software*, estão a janela de diagnóstico e a barra de status. Na janela de diagnóstico, é possível acompanhar todos os eventos que ocorrem na edição e teste da IHM, como variáveis alteradas, elementos criados, operações realizadas, entre outros. A barra de status mostra-se fundamental para a definição da clareza e consistência da IHM, pois, como as ferramentas de alinhamento, posicionamento e dimensionamento são muito simples, muitas vezes essas configurações tem que ser realizadas manualmente de acordo com os valores de posição e dimensão mostrados na barra de status.

6.2 Montagem do servidor e ambiente de simulação

Para desenvolver e testar a IHM é necessário que a mesma esteja conectada com a fonte de variáveis que serão monitoradas e manipuladas. Como não era possível ter acesso a aplicação real para desenvolvimento e teste da IHM, a solução encontrada foi a simulação da aplicação do caso em estudo. A simulação que foi utilizada para estudo do programa abordado no capítulo 4.2 só necessitou de um *link* de comunicação para integração com a IHM desenvolvida.

Para montar o ambiente de simulação, foram utilizados quatro softwares: *RSLogix Emulate* 500, *RSLogix* 500, *Factorytalk View* e *RSLinx*. A Figura 6.2 esclarece a relação e funcionalidade de cada ferramenta computacional no desenvolvimento da interface.

Figura 6.2 – Relação entre os softwares utilizados.

Download do programa para o CLP virtual

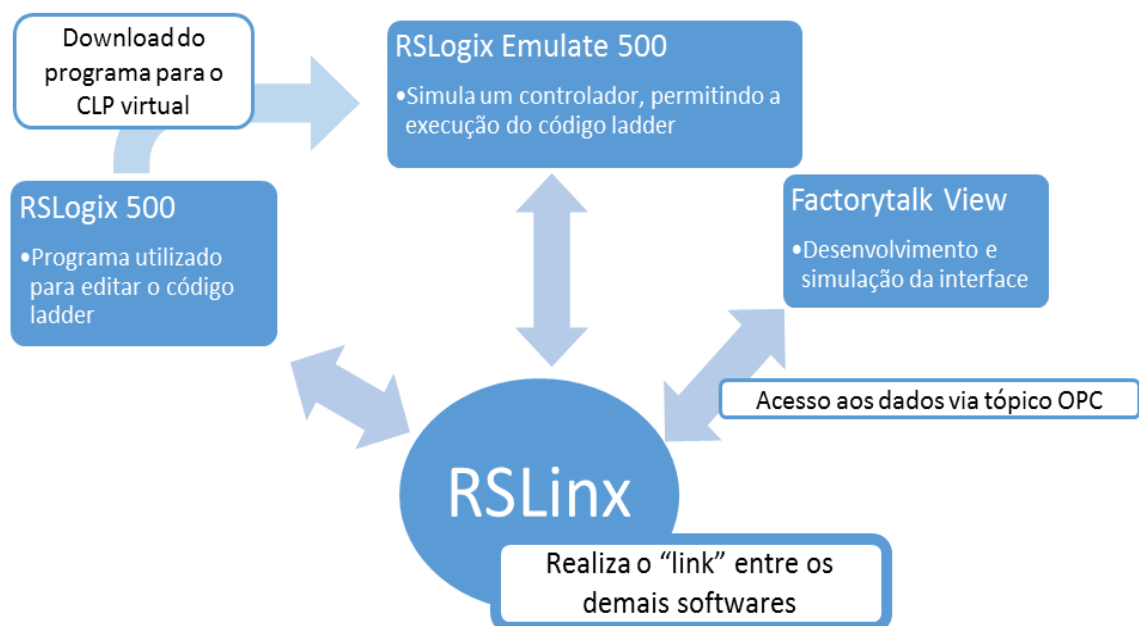
RSLogix Emulate 500
•Simula um controlador, permitindo a execução do código ladder

RSLogix 500
•Programa utilizado para editar o código ladder

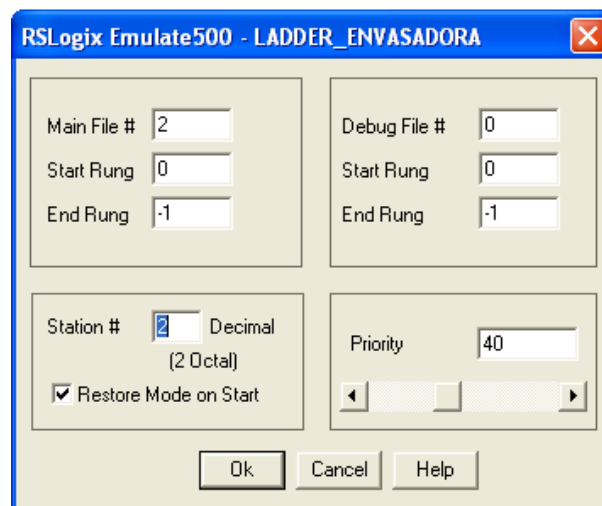
Factorytalk View
•Desenvolvimento e simulação da interface

Acesso aos dados via tópico OPC

RSLinx

Realiza o "link" entre os demais softwares

Fonte: Ilustração própria.

O *software RSLogix Emulate 500* desempenhou a função de simular um controlador *MicroLogix* 1500 (o mesmo utilizado no caso real). Para simular o CLP citado, a configuração é simples: basta carregar o arquivo *ladder* desejado e configurar parâmetros de execução e comunicação, como pode ser visto na Figura 6.3.

Figura 6.3 – Configuração do CLP simulado.

Fonte: Ilustração própria obtida com o auxílio do *software* RSLogix Emulate 500.

Como pode ser observado na Figura 6.3, o CLP simulado foi carregado com o programa *ladder* “LADDER_ENVASADORA” (programa apresentado no capítulo 4.2). O comportamento do controlador é definido de acordo com o arquivo principal e arquivo debug informados durante a configuração. No caso, o arquivo principal de execução era o *ladder* 2 do projeto carregado. Para concluir a configuração do CLP, foi definido o número da estação utilizada na rede virtual e selecionado modo *run* para garantir a execução do programa em tempo real.

O *software RSLinx* desempenha a função de estabelecer comunicação entre dispositivos de campo suportados pela *Rockwell* e os softwares da mesma. A configuração da comunicação é feita após a adição do *driver* adequado para o controlador simulado. O *driver* utilizado foi o “SLC 500 (DH485) Emulator Driver” que permite comunicação com todos os dispositivos simulados pelo *software RSLogix Emulate* 500. Após o startup do driver configurado, o controlador simulado já pode ser visto pelo computador. Apesar do dispositivo já estar online nessa etapa da configuração, a comunicação com o *software Factorytalk View* em um ambiente simulado deve ser feita por meio de um tópico OPC.

A configuração de um tópico OPC é feita por meio do *RSLinx*. Para abrir a tela de configuração, deve-se selecionar o dispositivo que será utilizado como fonte de dados por meio de um clique com o botão direito e seleção da opção “Configurar um novo tópico OPC”. Na tela de configuração do tópico, deve ser informado o tipo de processador, a frequência das mensagens tipo *polled*, o *driver* de comunicação, número da estação, entre outros parâmetros que podem ser utilizados com informações padrões. A Figura 6.4 mostra o tópico “OPCIHM” configurado para desenvolvimento da IHM em um ambiente de simulação.

Por fim, para concluir a configuração do ambiente de simulação utilizado, era necessária alguma forma de alterar o estado das variáveis da fonte de dados, pois, apesar da existência de um CLP simulado, os componentes sensores reais do caso em estudo não estavam conectados ao sistema. Para simular o comportamento do sistema e alterar variáveis, foi utilizado o *software RSLogix* 500. A simulação foi feita de forma manual, alterando parâmetros e bits específicos para reproduzir situações esperadas. Para ter acesso às variáveis, foi necessário utilizar o *software RSLogix* 500 no modo *online*. Primeiramente, foi carregado no *software* de desenvolvimento o mesmo projeto utilizado no *software* de simulação. Em seguida, foi apontado o endereço do CLP simulado no arquivo “*controller properties*” do projeto. Após o endereçamento, basta selecionar a opção “*go online”* no canto superior esquerdo do *software*. Como já foi dito, no modo *online*, é possível acompanhar o status das variáveis em tempo real e realizar a alteração da lógica e parâmetros do programa. A Figura 6.5 mostra uma visão geral do *software* RSLogix 500 após a execução dos passos descritos. É possível notar o modo “remote run” selecionado em verde (indicando que o programa está online), o arquivo “*Controller Properties*” exibido no início da árvore de arquivos do projeto e a janela de configuração do arquivo preenchida com os mesmos parâmetros do driver do CLP simulado definido no no *software RSLinx*.

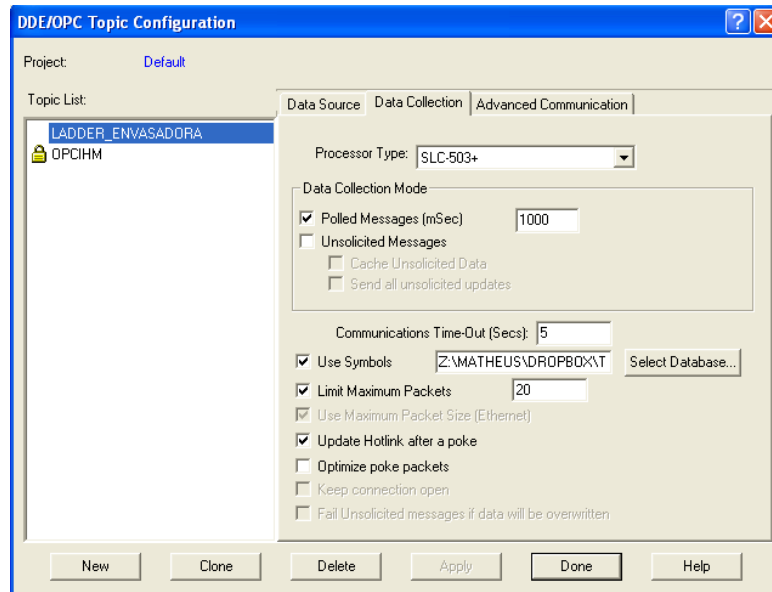
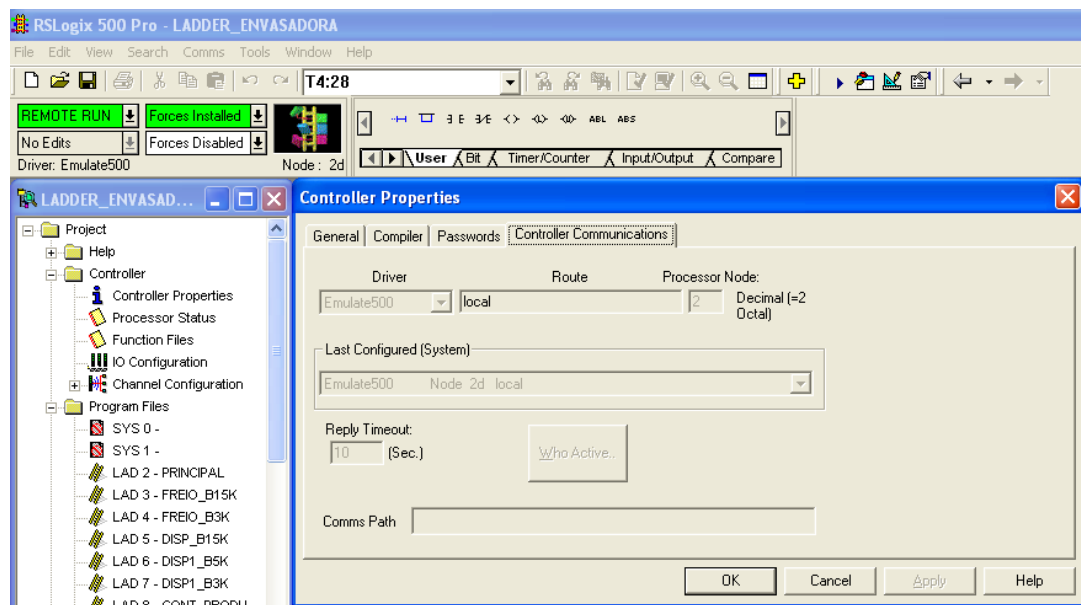
Figura 6.4 – Configuração de um tópico OPC no *software RSLinx*.

Fonte: Ilustração própria obtida com o auxílio do *software RSLinx*.

Figura 6.5 – Configuração do arquivo *ladder* em modo *online*.

Fonte: Ilustração própria obtida com o auxílio do *software RSLogix* 500.

6.3 Criação da IHM e recursos utilizados

Após o *startup* do ambiente de simulação, todos os requisitos para o desenvolvimento da IHM estavam satisfeitos. A programação da IHM utilizada neste trabalho foi baseada na interface real utilizada no caso em estudo, isto é, os principais parâmetros e comandos utilizados em uma IHM já implementada de uma máquina de envase foram incorporados no desenvolvimento desta interface simulada. Como melhoria, o projeto desenvolvido neste trabalho é adequado para um modelo de interface mais moderna, possibilitando o controle de tela por toque e uma tela colorida e maior.

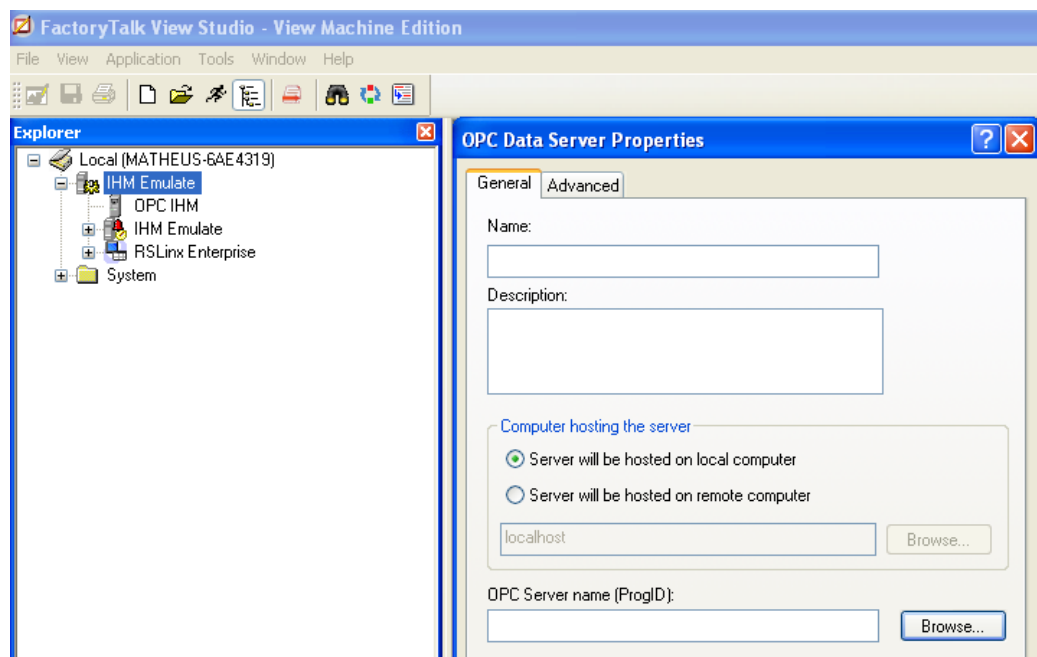
6.3.1 Criação da aplicação no Factorytalk View ME

O primeiro passo no desenvolvimento da IHM, é a criação do projeto no *software Factorytalk View*. Ao iniciar o aplicativo, deve ser escolhido o ambiente de trabalho *Machine Edition* por motivos já citados. No menu *File*, é possível escolher a opção *New Application* (Nova aplicação) e definir o nome do projeto. O *software* criará todos os elementos e bibliotecas que podem ser utilizadas no novo projeto. A primeira configuração a ser feita na aplicação, é a definição do tamanho da tela utilizada e, consequentemente, o modelo da interface correspondente ao tamanho de tela escolhido. Neste projeto, optou pela dimensão da tela de 640x480 pixels correspondente às interfaces *Panelview Plus 700* e *Panelview Plus 1000* da fabricante *Allen-Bradley* (Figura 6.6). Essa escolha é baseada na possiblidade de exibir várias informações de forma clara e pela praticidade de comando em telas sensíveis ao toque. A dimensão de tela escolhida permite a representação de informações que utilizavam 6 telas na interface real utilizada no caso em estudo em uma única tela da interface simulada.

Figura 6.6 – Modelo utilizado no caso real (esquerda) e modelo correspondente à interface simulada (direita).

Fonte: Allen-Bradley Graphic Terminals. Disponível em < http://ab.rockwellautomation.com/pt/Graphic-Terminals/2711-PanelView-Standard-300-Micro-Terminals>. Acesso em 15 de Maio de 2014.

Após essas configurações básicas, é necessário disponibilizar para a aplicação uma fonte de dados com variáveis que possam ser manipuladas. Esse passo é concluído com a adição do servidor OPC previamente configurado à aplicação. Para isto, basta clicar com o botão direito no ícone de projeto da janela de exploração e selecionar o item *Add New servers* (Adicionar servidor) e, em seguida, escolher a opção *OPC Data Server*. Na janela de configuração de servidor OPC a ser adicionado, deve ser criado o nome do servidor, informar em que computador está o servidor (deve ser informado a opção *localhost* no caso do servidor hospedado na mesma máquina utilizada para desenvolvimento da IHM) e a identificação do programa que está fornecendo o servidor OPC (foi escolhida a opção *RSLinx OPC Server*). Após essa configuração, já é possível utilizar as funções de leitura e escrita do tópico OPC configurado previamente. A Figura 6.7 mostra o servidor OPC utilizado neste projeto ("OPC IHM") já configurado ao lado esquerdo da tela de adição de um novo tópico.

Figura 6.7 – Adição de um servidor OPC no *software Factorytalk View ME*.

Fonte: Ilustração própria obtida com o auxílio do *software Factorytalk View ME*.

6.3.2 Desenvolvimento gráfico da IHM

Após todas as configurações feitas para estabelecer um ambiente de simulação de um projeto de automação e possibilitar o acesso da IHM às variáveis simuladas, iniciou-se a etapa de desenho da interface gráfica. O *software Factorytalk View ME* possui diversos objetos gráficos que permitem ao usuário final da aplicação uma operação mais amigável. Uma IHM é composta basicamente por telas que agrupam elementos gráficos que manipulam

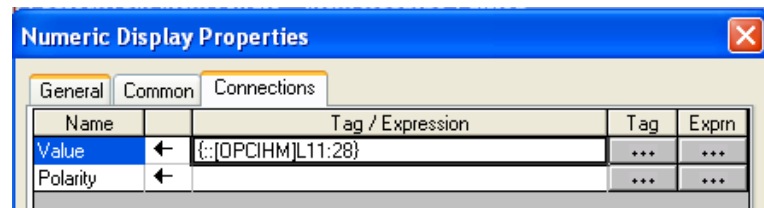
as variáveis do processo. Para o completo entendimento do desenvolvimento do processo, é necessário ter noções básicas dos recursos gráficos disponíveis no *software Factorytalk View ME*.

O primeiro elemento gráfico utilizado é o *display* ou "tela". É esse elemento que permite a criação de diversos ambientes com informações relacionadas para facilitar a operação. Um dos parâmetros configuráveis do *display* é o seu tipo, que pode ser escolhido como *replace* ou *on top*. Na opção *replace* ou substituição, quando o *display* for requisitado pelo usuário, ele substituirá a tela atual por completo, por isso suas dimensões estão de acordo com as medidas originais determinadas no início do projeto. Já no modo *on top* ou "no topo", o *display* será posicionado por cima da tela atual e pode ter o seu posicionamento e tamanho especificado. Além do seu tipo, cada *display* pode receber um código de segurança e um número de identificação que terão suas funcionalidades explicadas nas próximas seções deste trabalho.

Após a criação de um *display*, cabe ao programador determinar quais elementos gráficos serão utilizados para interagir com o usuário final. O programa apresenta elementos mais básicos que não requisitam muito conhecimento para serem utilizados e podem ser criados de forma intuitiva, como textos e figuras geométricas. Para o desencadeamento de ações ou alteração de variáveis booleanas, os elementos utilizados são os botões. O programa oferece os mais diversos tipos de botões: momentâneo, retentivo, controle de rampa, entre outros. Esses botões podem ser desenhados livremente ou ter suas dimensões precisamente especificadas na aba *Common* do painel de configuração do elemento. Os parâmetros gráficos modificáveis deste elemento são: cor de fundo, tipo de fundo, borda e texto mostrado. Esses atributos gráficos podem ter uma configuração específica para cada estado (ativado ou desativado). Os parâmetros funcionais do objeto são configurados na aba *Connections*, onde os botões serão associados com variáveis do processo de automação.

A maioria dos elementos gráficos da IHM possibilita a definição de conexão com parâmetros do projeto de automação. Essas conexões geralmente são especificadas no campo *Caption* ou na aba *Connections*. O campo *Caption* define o que será mostrado em um objeto que permite a exposição de texto, como o texto que será mostrado em um painel de acordo com uma variável do tipo *string* associada ao campo *Caption*. Já a aba *Connections* permite a alteração de uma variável de acordo com uma ação tomada na IHM, como a troca do valor de uma *tag* após um clique no botão. Ambos os campos de conexões permitem o uso de expressões para manipular os valores da *tag*. A Figura 6.8 mostra uma *tag* do servidor OPC

apontando pro arquivo 11 (elemento 28) de variáveis do tipo *long* do projeto de automação simulado. O sentido da seta indica o tipo de conexão (seta voltada para esquerda simbolizando uma conexão de leitura)

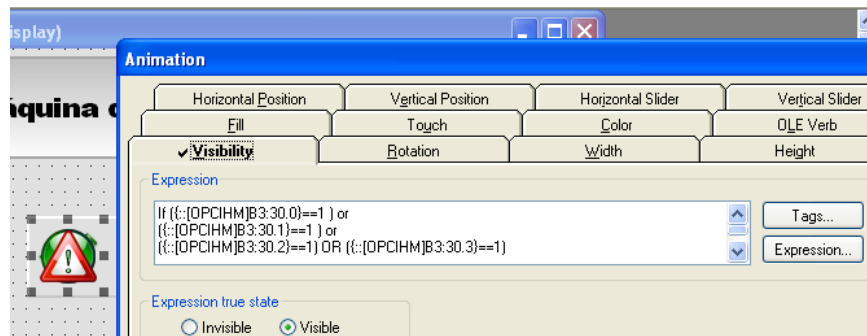
Figura 6.8 – Configuração das conexões de objetos gráficos.

Fonte: Ilustração própria obtida com o auxílio do *software Factorytalk View ME*.

A Figura 6.8 mostra parte do painel de configuração de um outro tipo de elemento: os *displays* de valores. Na IHM desenvolvida os mostradores de valores foram amplamente utilizados, pois são esses elementos que permitem o acompanhamento e alteração de valores do tipo inteiro. Os mostradores de valores numéricos podem ser encontrados em dois tipos: *Numeric Display* ou *Numeric Input Display*. O primeiro apenas mostra valores. Já o segundo permite a alteração do valor mostrado após um clique no mostrador.

Outro recurso bastante aplicado no desenvolvimento desta IHM foram as imagens. O uso de imagens facilita o reconhecimento de status e padrões pelo usuário final. O programa *Factorytalk View* oferece uma biblioteca (*Symbol Factory*) de símbolos comumente usados no controle de processos. Porém, esses símbolos não podem ser aplicados diretamente à um *display*. Todas as imagens devem ser salvas e, em seguida, importadas para a biblioteca de imagens do programa. Apenas as figuras contidas na biblioteca de imagens podem ser utilizadas nos *displays*, botões, entre outros. Esse requisito torna o uso de imagens nas interfaces bastante burocrático, incentivando a aplicação de gráficos nativos do *software*.

Vários dos recursos citados neste tópico não seriam realmente eficientes no uso final de uma interface se não fosse pela funcionalidade proporcionada pelas animações. Sem dúvidas, uma IHM eficiente deve contar com elementos dinâmicos, que se alterem de acordo com o status observado em um processo. Nas últimas etapas do desenvolvimento gráfico da IHM, entre em ação as animações, isto é, após adicionar-se uma imagem para cada status de uma determinada variável, é interessante que apenas a imagem associada ao estado atual seja mostrada. Cada elemento inserido em uma tela, possui diversas opções de animação, como visibilidade, rotação, preenchimento, cor, entre outros. A principal animação utilizada neste projeto de interface foi a visibilidade. Com essa animação é possível tornar um objeta invisível durante a execução da interface de acordo com o valor de uma variável monitorada.

Na configuração de uma animação, uma variável deve ser atribuída com ou sem uma expressão. Em seguida, é escolhido se o elemento será visível quando a expressão for verdadeira ou falsa. A mostra várias imagens sobrepostas ao lado da sua respectiva janela de configuração de animações. Pode-se ver que uma expressão lógica está associada a diversas *tags* que definem a visualização de apenas uma imagem durante a execução da IHM.

Figura 6.9 – Configuração da visibilidade de um elemento

Fonte: Ilustração própria obtida com o auxílio do *software Factorytalk View ME*.

Por fim, para encerrar a abordagens sobre recursos utilizados no desenvolvimento gráfico da interface, é necessário abordar o funcionamento dos *Global Objects* ou objetos globais. Muitas vezes uma interface necessita da presença de um grupo de elementos gráficos em várias telas, como na criação de um cabeçalho ou painel de navegação. Para permitir uma flexibilidade na alteração desses grupos de objetos gráficos, esses objetos são criados de forma global. Os objetos globais são criados separadamente e copiados para as demais telas do projeto de interface. A cópia dos objetos globais em cada *display* permanece associada ao objeto de origem, isto é, basta alterar o objeto original para que todas as suas cópias sejam atualizadas com as devidas modificações. No projeto desenvolvido neste trabalho, o cabeçalho da interface e os painéis das tabelas de parâmetros foram criados a partir de um objeto global.

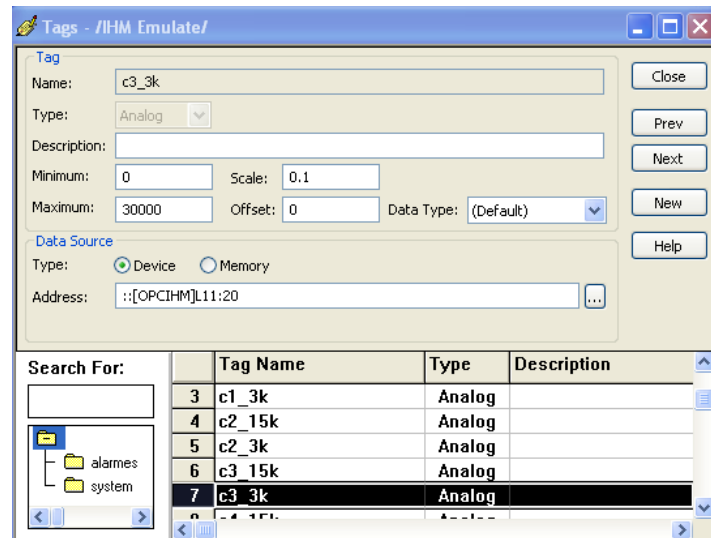
6.3.3 Configuração de tags

Após o desenvolvimento da parte gráfica da IHM, deve ser definido como a interface interagirá com a fonte de dados. Essa função está diretamente relacionada com as *tags*. As *tags* são nomes lógicos atribuídos as variáveis internas ou externas da IHM que podem facilitar a manipulação de dados no projeto. Seja a aplicação desenvolvida em um computador pessoal ou em um *diplay Panelview*, a IHM tem capacidade de armazenar variáveis internamente, isto é, podem ser criadas variáveis para controle de alguma lógica interna da interface sem interferir com os dispositivos externos. Como já foi abordado, a fonte

externa de dados do projeto consistia em um servidor OPC, mas essa fonte poderia ser fornecida por uma rede real ou ligação direta com o controlador lógico.

Na maior parte do projeto, foi utilizado o endereçamento direto para estabelecer conexões entre os elementos gráficos e as variáveis do processo por motivos de praticidade e performance, pois a criação de uma *tag* para cada variável manipulada poderia resultar em uma queda de desempenho da interface. Porém, em algumas situações foi necessário o uso de *tags* para fonte de dados externa e interna. É aconselhável o uso de *tags* internas quando é necessário utilizar um fator de escala ou de *offset* em uma fonte de dados. Muitas vezes o valor manipulado pelo controlador não condiz com o valor real utilizado no processo. Por exemplo, uma temperatura pode passar por uma série de operações matemáticas ou conversão analógica/digital e ser armazenada em um arquivo de dados do controlador. Esse valor registrado será diferente do valor observado no processo, resultando em uma situação não muito clara para operação do usuário final. Nesta situação, podem ser utilizadas *tags* da IHM para realizar a conversão de escala e impor valores máximos e mínimos para o operador sem interferir na lógica da aplicação executada no CLP.

Para criar novas *tags,* deve ser utilizado o recurso *Tags Editor* do *software Factorytalk View*. Os parâmetros configuráveis nesse editor variam de acordo com o tipo da *tag* criada: analógico, digital ou *string*. As *tags* analógicas permitem a definição de um valor máximo e mínimo, de um *offset* e de um fator de escala. Além disso, pode ser definido o tipo de dado: inteiro, *long*, *byte*, entre outros. As *tags* do tipo digital só permitem a configuração do valor inicial (0 ou 1) e a característica retentiva para fonte de dados externas. A *tag* do tipo *string* permite apenas a definição do número de caracteres. As configurações comuns a todos os tipos de *tags* são: descrição e fonte de dados dispositivo (requer um endereço de um CLP ou servidor OPC) ou memória (*tag* armazenada apenas na interface). A Figura 6.10 mostra a tela de edição das *tags* da IHM.

Figura 6.10 – Editor de *tags* do *software Factorytalk View ME.*

Fonte: Ilustração própria obtida com o auxílio do *software Factorytalk View ME*.

A maioria das *tags* utilizadas foram do tipo analógico, pois esse tipo permite o uso de um fator de escala para valores inteiros, facilitando, por exemplo, a leitura dos parâmetros de *setpoint* da pesagem das balanças da aplicação. A Tabela 6.1 lista todas as *tags* utilizadas no projeto.

Tabela 6.1 – *Tags* utilizadas no desenvolvimento d IHM

Tag	Tipo de Tag	Fonde de dados	Descrição
c1_15k	Analógica	L11:10	Setpoint célula 1 - 15 Kg
c2_15k	Analógica	L11:11	Setpoint célula 2 - 15 Kg
c3_15k	Analógica	L11:12	Setpoint célula 3 - 15 Kg
c4_15k	Analógica	L11:13	Setpoint célula 4 - 15 Kg
pesoantecipacao_15k	Analógica	L9:31	Peso de antecipação - 15 Kg
c1_3k	Analógica	L11:18	Setpoint célula 1 - 3 Kg
c2_3k	Analógica	L11:19	Setpoint célula 2 - 3 Kg
c3_3k	Analógica	L11:20	Setpoint célula 3 - 3 Kg
c4_3k	Analógica	L11:21	Setpoint célula 4 - 3 Kg
pesoantecipacao_3k	Analógica	L9:33	Peso de antecipação - 3 Kg
tela	Analógica	Interna	Define a tela a ser exibida
ligamaquina	Digital	B3:1/0	Liga a máquina envasadora
balanca	Analógica	Interna	Define a balança monitorada

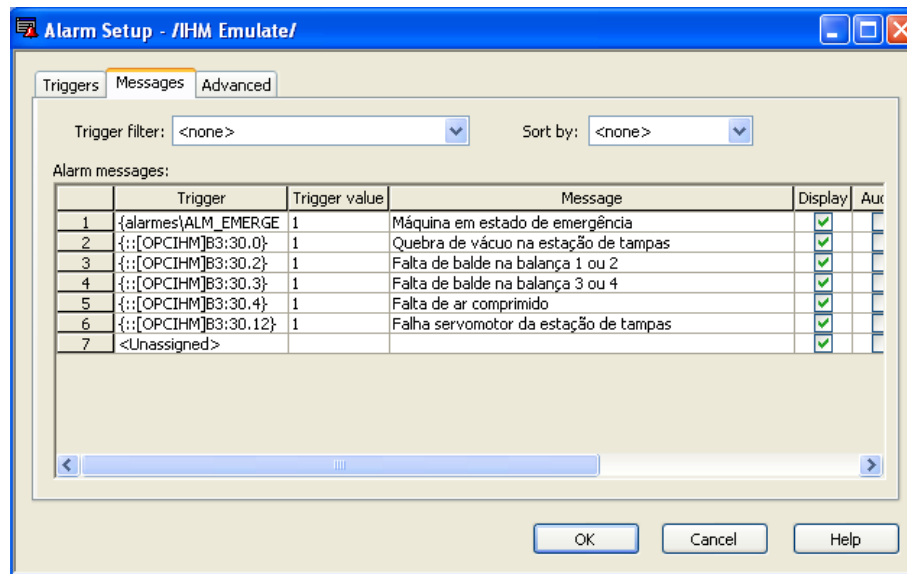
Fonte: Tabela própria.

6.3.4 Configuração de alarmes

Durante a supervisão de um processo, o operador responsável pela planta/máquina deve saber a existência de uma situação de risco de modo instantâneo. Essa funcionalidade é alcançada com o uso de alarmes que conseguem identificar a falha ou proximidade de uma condição perigosa em um dispositivo ou processo.

Para atender à necessidade desses alertas instantâneos, o *software Factorytalk View* conta com uma ferramenta para configuração de alarmes. Primeiramente, na tela de configuração de alarmes, devem ser informados os gatilhos dos alertas, isto é, variáveis que serão monitoradas e, ao alcançarem um valor de risco definido pelo programador, irão desencadear ações e alertas instantâneos. A definição de um gatilho pode ser feita por meio de uma referência a uma variável interna ou endereçamento direto. Como o programa do CLP já contava com bits específicos para situações de risco, os gatilhos foram referenciados aos bits do programa *ladder* de alarmes do CLP. Após a definição das variáveis relacionadas aos gatilhos do alarme, deve-se definir qual de cada variável acionará o alarme e como esse alarme deve ser processado. Em uma condição de falha, é possível habilitar a visualização de um *display*, o envio de uma mensagem para uma impressora ou o envio de uma mensagem para um dispositivo remoto (CLP). No caso da IHM desenvolvida, foi feita a escolha de exibir mensagens de alerta durante a ocorrência de uma falta. Como as variáveis escolhidas como gatilho eram do tipo digital, o valor verdadeiro foi escolhido como condição de disparo do alarme. Também, para cada variável foi definida uma mensagem a ser exibida. Por fim, na aba de configurações avançadas, foi apontado o *display* “alarmes” para ser exibido no caso do disparo de algum gatilho. A Figura 6.11 mostra a tela de configuração dos alarmes, onde é possível observar as mensagens definidas e o valor de disparado para cada gatilho. É importante ressaltar que, para o funcionamento de todas essas definições de alarmes realizadas, deve ser marcada a caixa de seleção *Alarms* no arquivo de configuração *Startup* encontrado na área de navegação.

.

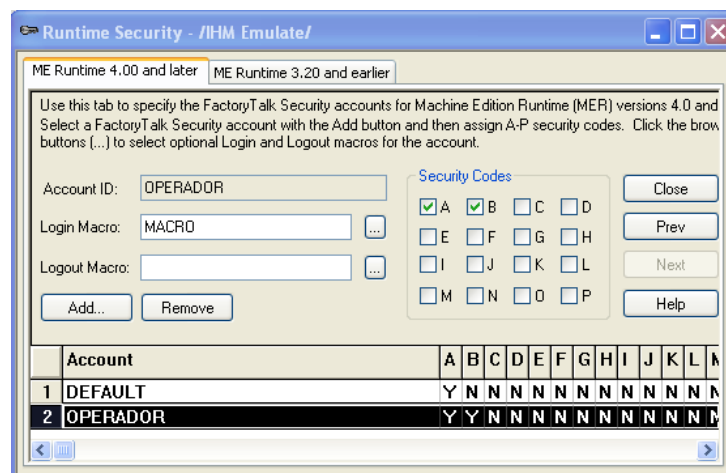
Figura 6.11 – Editor de alarmes do *software Factorytalk View ME.*

Fonte: Ilustração própria obtida com o auxílio do *software Factorytalk View ME*.

6.3.5 Segurança, macros e Global Connections

O ponto final do desenvolvimento da interface simulada neste trabalho se deu na criação de um sistema de segurança. Uma interface que permite a alteração dos parâmetros mais importantes de uma máquina ou processo não pode permitir o acesso de qualquer usuário. Esse limite de acessibilidade é muito comum na utilização de IHMs. Por isso, o *software Fatorytalk View* conta com ferramentas capazes de definir os tipos de usuários que irão interagir com a interface e quais os elementos do projeto estão associados a cada tipo de usuário.

O primeiro passo para configurar um sistema de segurança, é criar os usuários e grupos no sistema da aplicação. Para isso, na pasta *System*, subpasta *User and Groups*, podem ser adicionados novos usuários. O nome do usuário é uma informação obrigatória. Como informações opcionais, pode ser informada uma senha ou selecionar uma opção que permita que o usuário defina sua senha em seu primeiro *login*. Para facilitar a configuração do nível de acesso em sistema com muitos usuários, pode-se optar pela utilização de grupos. Desta forma, os níveis de acesso podem ser definidos para um grupo de usuários que têm o mesmo nível de acessibilidade ao invés de configurar o acesso para cada usuário. Os grupos são configurados na mesma pasta *System* com o nome do grupo, descrição e membros participantes. Por padrão, um projeto de interface no *Factorytalk View* já conta com um usuário *DEFAULT* tem acesso aos elementos que não necessitam de nenhum nível de segurança. Por isso, no projeto desenvolvido, só foi adicionado mais um usuário: o “operador”.

Após a criação dos usuários e grupos, deve-se configurar quais privilégios cada usuário terá. Para isto, utiliza-se a ferramenta *Runtime Security*. Os níveis de acesso do programa são definidos pelos *Security Code* ou “códigos de segurança”. O programa conta com 16 códigos alfabéticos (letras de A à P). Os códigos de segurança fazem parte da configuração dos *displays* da IHM, isto é, as telas da interface vêm configuradas com o caractere “*” no campo de código de segurança (o que permite o acesso de qualquer usuário a tela), mas pode ser atribuído um código especifico para cada tela, fazendo que só usuários com aquele nível de acesso abram a tela. Ao adicionar um usuário ao sistema de segurança da IHM, deve ser definida à qual letra o usuário terá acesso. A Figura 6.12 mostra a tela de configuração da ferramenta *Runtime Security*.

Figura 6.12 – Definições de segurança do *software Factorytalk View ME*.

Fonte: Ilustração própria obtida com o auxílio do *software Factorytalk View ME*.

Com a utilização de dois usuários apenas, definiu-se a utilização de dois níveis de acesso: A para o usuário comum e B para usuários autenticados. O projeto de IHM foi desenvolvido com todas as telas liberadas apenas para nível B, restando apenas uma tela inicial com nível A onde seriam exibidas algumas informações básicas e seria possibilitado o *login* de usuário para acessar a interface de forma completa.

Definidos os usuários e acessos a serem utilizados, era necessário desenvolver uma forma de iniciar em uma tela de acesso comum e, após efetuação do *login*, abrir o menu da IHM que permite o acesso a todas as telas da interface. A solução foi encontrada por meio da combinação de duas ferramentas: macros e *Global Connections*.

As conexões globais permitem a integração entre a fonte de dados, seja uma variável interna ou de um dispositivo externo, e variáveis de controle da IHM, isto é, por meio

destas conexões, datas, macros, telas atuais, entre outros parâmetros, podem ser alterados remotamente por um CLP. Neste projeto, particularmente, foi utilizada a conexão Remote Display Number. Esta conexão pode ser atribuída a uma tag que informará o número da tela a ser exibida na IHM. Cada display, além de um código de segurança, possui um número identificador. Ao optar pelo controle remoto da tela exibida, é necessário ter um cuidado adicional com a fonte de dados no controle, pois com este tipo de conexão pode ser encontrada uma brecha de segurança, isto é, a tela informada, por meio do seu número, pela tag de controle remoto será exibida independente do seu nível de segurança e do usuário logado. Quando um valor diferente de zero é informado a conexão de controle remoto de tela, o usuário local perde o controle sobre a navegação da interface, isto é, quando um valor é informado à conexão citada, a navegação entre telas passa a ser exclusivamente remota até o retorno da conexão ao valor zero.

Assim, foi atribuída a tag “tela” à conexão Remote Display Number. Porém, era necessário criar uma forma de controle desta variável interna, pois ela deveria controlar a navegação somente durante o logon do usuário e, em seguida, devolver o controle de telas ao usuário já autenticado. Para alcançar este objetivo, foi necessária a utilização de macros.

Uma macro é uma lista de definições de tags que é executada eventualmente, isto é, diversas ações da IHM permitem a associação de um macro, resultando na definição de uma lista de valores após a ocorrência de um evento (startup da aplicação por exemplo). Ao criar uma macro, o usuário pode informar uma lista de tags e suas respectivas expressões ou valores. Em seguida, esse arquivo pode ser executado em diversas situações de uma aplicação do *software* Factorytalk View, como inicialização da interface, inicialização de uma tela, logon de um usuário, entre outros. Para controle dos acessos de usuários, foram criados dois arquivos de macros: “Logon” e “Menu”. A primeira macro (Logon) foi associada ao botão de login da interface, isto é, seria executada sempre que houver o logon de um usuário na aplicação. Esta macro modifica a variável interna “tela” (associada ao controle remoto de displays) para o valor 20, que é o valor da tela de menu para usuários autenticados. Já a macro “Menu” está associada a inicialização do display “menu” da IHM. Quando executada, esta macro atribui o valor 0 para a variável interna “tela”, devolvendo o controle da navegação de telas para o usuário local.

Resumidamente, o processo de logon ficou da seguinte forma: uma variável interna está associada ao controle remoto de telas. Não é possível habilitar o controle remoto e local ao mesmo tempo. Portanto, quando a conexão de controle remoto de tela possui um

valor diferente de zero, a tela exibida é igual ao valor informado e, quando o valor da conexão for zero, o controle de telas é realizado pelo operador da IHM manualmente. Ao realizar um logon, é informado o valor de tela 20, correspondente ao menu próprio para usuário autenticados. Ao abrir esse menu, outra macro é executada, definindo o valor da conexão remota para zero (controle local).

6.4 Resultados e funcionamento da IHM

Após abordar todos os recursos utilizados no desenvolvimento da IHM simulada, esta seção do trabalho apresenta o resultado final obtido. Serão apresentadas todas as telas desenvolvidas e os seus respectivos princípios de funcionamento. Ao todo, foram desenvolvidos 11 displays no projeto, abrangendo um bom número de funcionalidades do *software* Factorytalk View e controlando diversos parâmetros da automação de uma máquina envasadora estudada durante este trabalho.

6.4.1 Display “inicial”

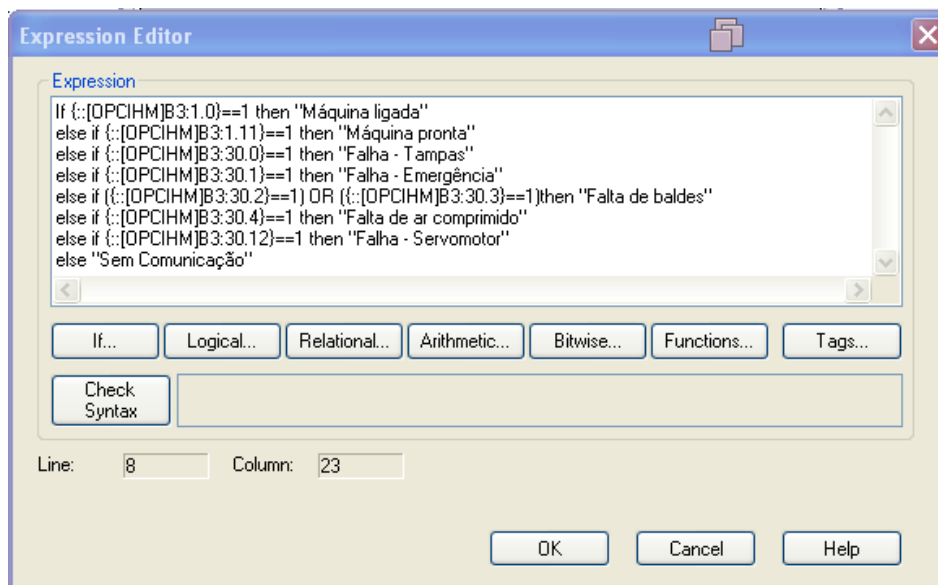
O primeiro *display* utilizado foi o de nome “inicial”. Esta tela está associada com as configurações de *startup* da aplicação da interface, ou seja, é a primeira tela a ser exibida e possui código de segurança “A” (todos usuários têm acesso). A Figura 6.13 mostra o *display* “inicial” em execução. Na parte superior da tela, está o cabeçalho utilizado na maioria dos *displays*, com a única diferença da ausência do botão de navegação para o menu, pois o menu só é permitido a usuários autenticados. O *display* apresenta informações básicas para o usuário comum, como número de baldes produzidos pela máquina e o atual status da mesma. A informação do estado da máquina é obtida por meio de uma expressão que monitora diversas *tags* do controlador e define mensagens de acordo com os valores encontrados. A Figura 6.14 mostra a expressão utilizada para exibir a mensagem de status.

Abaixo da tabela de informações básicas, encontram-se a duas únicas ações permitidas ao usuário comum: *login* e acionar a emergência da máquina. O botão de *login* chama um teclado próprio para telas sensíveis ao toque que permite a entrada de um nome de usuário e uma senha. Após o *login,* as funcionalidades da IHM são liberadas. O acionamento da emergência foi implementado por meio de um botão retentivo. Em cima do botão foram usados dois mostradores de texto e duas imagens para facilitar o entendimento do usuário. Nesses elementos, foi aplicada uma animação de visibilidade para que apenas uma mensagem

de texto e uma figura fossem exibidas de acordo com o estado de emergência da máquina indicado pelo bit “B3:30.1” do controlador.

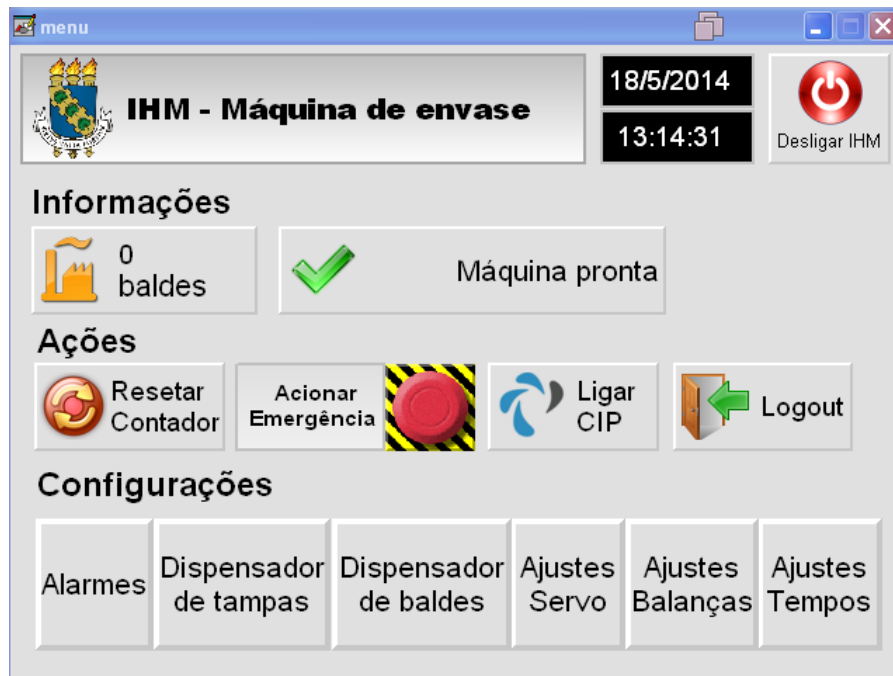
Figura 6.13 – Tela “inicial” da aplicação desenvolvida.

Fonte: Ilustração própria obtida com o auxílio do *software Factorytalk View ME*.

Figura 6.14 – Expressão do status da máquina.

Fonte: Ilustração própria obtida com o auxílio do *software Factorytalk View ME*.

6.4.2 Display "menu"

Após a realização do *logon* de um usuário, todas as telas acessadas possuem código de segurança "B" (somente o usuário "operador" tem privilégio de acesso "B"). A primeira tela exibida para um operador autenticado foi nomeada "menu" (Figura 6.15). Nesta tela, o operador pode observar as mesmas informações da tela "inicial", navegar para outras telas de configuração e realizar operações básicas, como resetar o contador de produção, acionar o botão de emergência da máquina, ligar o sistema de limpeza da máquina (CIP – *Clean in place*) e efetuar o *logout* do sistema. No cabeçalho, o botão de navegar para o menu utilizado nas demais telas foi substituído por um botão de desligamento da interface.

Figura 6.15 – Tela "menu" da aplicação desenvolvida.

Fonte: Ilustração própria obtida com o auxílio do *software Factorytalk View ME*.

Todos os elementos da linha "configurações" da tela "menu" são botões de navegação. A configuração desse tipo de objeto consiste no apontamento do arquivo *display* a ser carregado após o clique do botão. O painel de status da máquina conta com quatro figuras sobrepostas que são controladas por visibilidade de acordo com uma das seguintes situações: máquina pronta para ser ligada, máquina ligada, interface sem comunicação com o CLP e estado de falha. A Figura 6.16 demostra essa animação aplicada no status da máquina. A mensagem de falha varia de acordo com o alarme disparado na falha.

Figura 6.16 – Animação painel de status da máquina

Fonte: Ilustração própria obtida com o auxílio do *software Factorytalk View ME*.

Ao clicar no botão *logout*, o usuário é alterado para "*default*" e a tela "inicial é exibida. A sessão do usuário também é finalizada no caso de cinco minutos de inatividade do operador.

6.4.3 Dispensador de baldes

A IHM desenvolvida conta com duas telas de configuração da estação do dispensador de baldes: "disp_baldes1" e "disp_baldes2". Essas telas contam com tabelas que permitem a exibição dos parâmetros atuais de ajuste do dispensador de baldes de 3 quilos e 15 quilos. Os valores mostrados nas duas telas também podem ser diretamente alterados. A Figura 6.17 e a Figura 6.18 mostram as telas citadas em execução e seus parâmetros no momento da simulação.

Figura 6.17 – Tela "disp_baldes2" da aplicação desenvolvida.

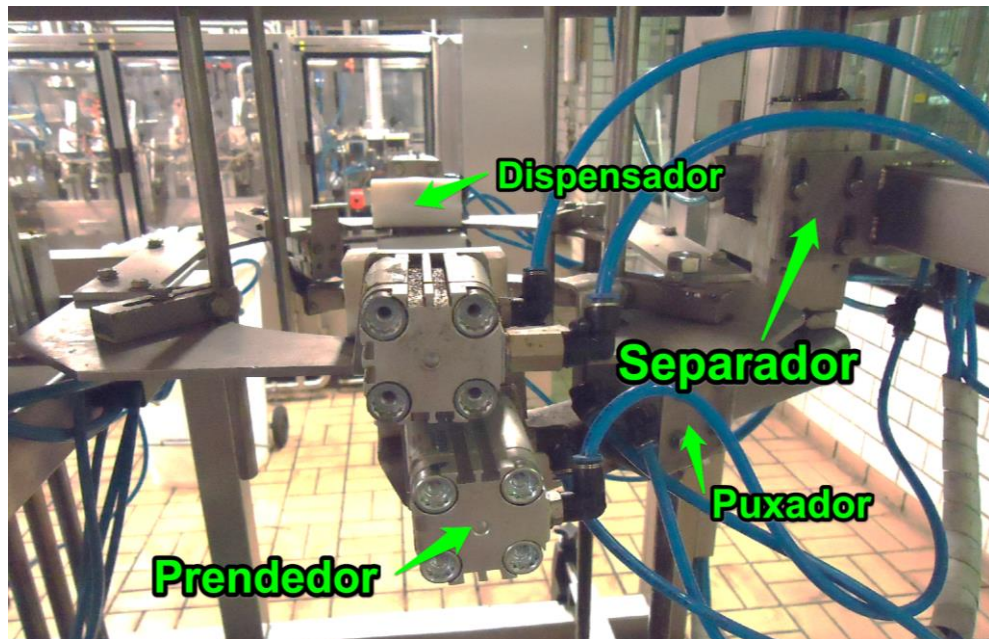
Fonte: Ilustração própria obtida com o auxílio do *software Factorytalk View ME*.

Figura 6.18 – Tela “disp_baldes1” da aplicação desenvolvida.

Fonte: Ilustração própria obtida com o auxílio do *software Factorytalk View ME*.

Como já foi abordado no capítulo 4 e na seção 5.3.3, o dispensador de baldes é o primeiro processo da máquina envasadora estudada, dispensando baldes na esteira de entrada da máquina. O dispensador de baldes conta com quatro atuadores pneumáticos: freio superior ou dispensador, prendedor, separador e puxador. Esses elementos podem ser observados na Figura 6.19. Resumidamente, foi implementado um temporizador cíclico virtual. Neste temporizador, um clock incrementa um ângulo virtual. O temporizador é composto por *cames*. Cada *came* aciona um componente do processo durante determinada faixa do ângulo virtual.

No dispensador implementado, cada um dos quatro atuadores pneumáticos é acionado dentro de uma faixa de um ângulo virtual, isto é, correspondem à uma *came* do temporizador cíclico virtualizado. Portanto, os valores exibidos nas tabelas dos *displays* do dispensador de baldes correspondem à um ângulo que varia de 0 à 360 graus. Nas telas mostradas cada atuador pneumático do dispensador de baldes pode ter o seu limite inferior e superior de ângulo virtual modificados, possibilitando que o operador ajuste durante qual período de um ciclo do dispensador de baldes cada atuador pneumático ficará acionado.

Figura 6.19 – Detalhe dos atuadores do dispensador de baldes.

Fonte: Ilustração própria.

6.4.4 Dispensador de tampas

Assim como os *displays* de configuração do dispensador de baldes, a estação do dispensador de tampas conta com duas telas de configuração dos seus parâmetros: “disp_tampas1” e “disp_tampas2”. A Figura 6.20 e a Figura 6.21 mostram as telas desenvolvidas em funcionamento durante uma simulação. Assim como no dispensador de baldes, as telas são compostas por tabelas desenhadas manualmente, isto é, cada linha, fundo e texto das tabelas tiveram que ser desenhados a partir de figuras básicas como retângulos e linhas, pois o *software Factorytalk View* não conta com um elemento de tabela que permita apenas a edição dos conteúdos de cada célula. Os valores mostrados são editáveis a partir de um toque na tela ou clique, chamando um teclado para entrada de valores. No canto inferior, há um botão que permite a navegação entre a primeira e segunda tela de configuração do dispensador de tampas.

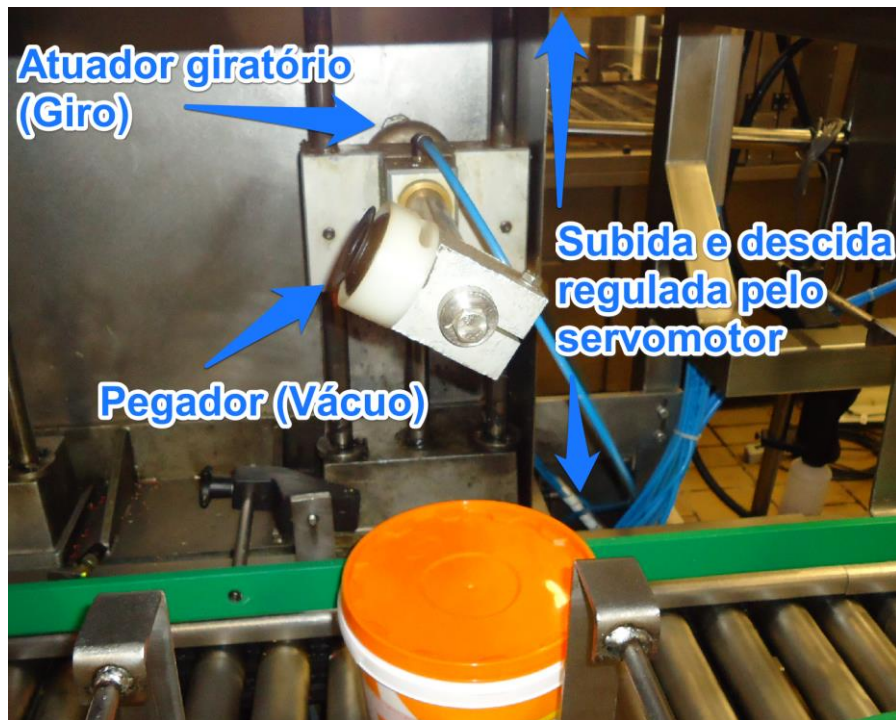
Figura 6.20 – Tela “disp_tampas1” da aplicação desenvolvida.

Fonte: Ilustração própria obtida com o auxílio do *software Factorytalk View ME*.

Figura 6.21 – Tela “disp_tampas2” da aplicação desenvolvida.

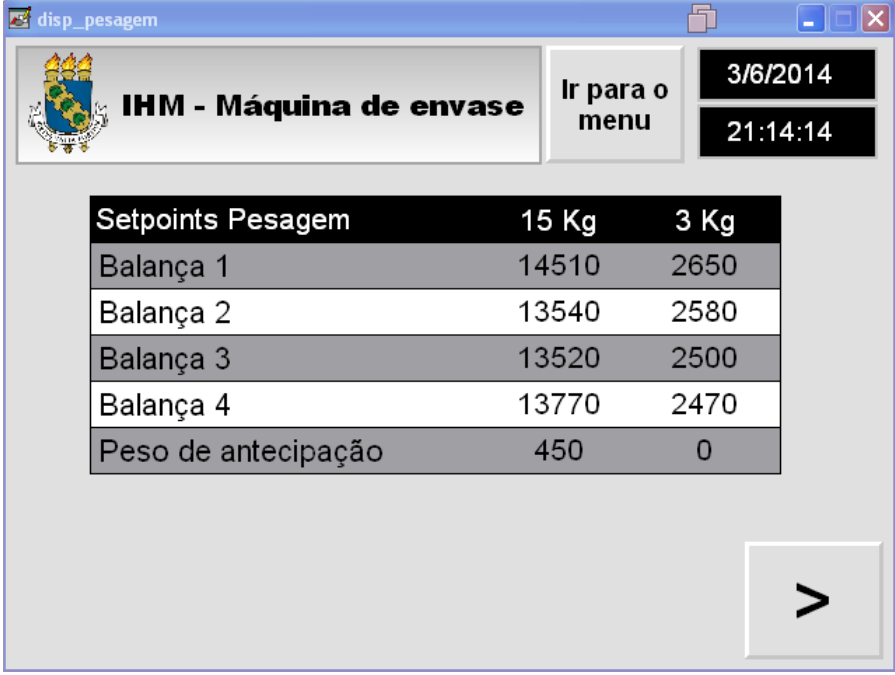
Fonte: Ilustração própria obtida com o auxílio do *software Factorytalk View ME*.

A solução de automação implementada no dispensador de tampas é a mesma do dispensador de baldes: um temporizador cíclico virtual. Porém, na estação de tampas o ciclo é composto por seis processos distintos, utilizando seis faixas de ângulos virtuais. Dos seis *cames* utilizados, quatro acionam atuadores, um testa a existência do vácuo e outro estabelece o tempo em que o dispensador fica parado. Os quatros atuadores acionados durante o ciclo de operação da estação de tampas são: ventosa a vácuo para pegar tampas, cilindro giratório responsável pelo giro do braço mecânico, servomotor responsável pelo posicionamento da tampa e dispensador de tampas (freio da pilha de tampas). Esses elementos atuadores podem ser observados na Figura 6.22.

Figura 6.22 – Detalhe dos atuadores do dispensador de tampas.

Fonte: Ilustração própria.

6.4.5 Display de ajuste de temporizadores

A IHM desenvolvida conta com uma tela própria para ajustes de algum dos principais temporizados da aplicação me *ladder* do caso em estudo. O *display*, nomeado "disp_tempos1", possui construção semelhante com as dos *displays* utilizadas para ajustes de parâmetros do dispensador de tampas e de baldes, isto é, conta com uma tabela preenchida com objetos do tipo *Numeric Input Display* que permitem a visualização dos valores atuais dos parâmetros e a chamada de um teclado para inserção de novos valores. A Figura 6.23 mostra *display* de ajuste de temporizadores em execução.

Figura 6.23 – Tela “disp_tempos1” da aplicação desenvolvida.

Fonte: Ilustração própria obtida com o auxílio do *software Factorytalk View ME*.

Os tempos escolhidos para edição do operador, são aqueles que não requisitam um conhecimento técnico sobre o desenvolvimento do código *ladder* da aplicação. Estão disponíveis para modificação os tempos de espera para ligar ou desligar as três esteiras da máquina, tempo para ligar o dispensador de baldes e ajuste do clock do temporizador cíclico virtual do dispensador de baldes.

6.4.6 Displays de ajuste dos parâmetros de pesagem

O ajuste dos parâmetros das quatro células de carga da máquina de envase automatizada é foi por meio de duas telas. O primeiro *display* disponível para ajuste das configurações da estação de pesagem foi nomeado “disp_pesagem”. Nesta tela, encontram-se uns dos parâmetros mais importantes para o ajuste do operador da máquina: os valores alvos das pesagens das quatro células de carga da máquina. O *display* segue o padrão implementado nas telas de configurações da IHM, isto é, apresenta os parâmetros ajustáveis por meio de uma tabela. A Figura 6.24 mostra o *display* citado durante simulação.

Figura 6.24 – Tela “disp_pesagem” da aplicação desenvolvida.

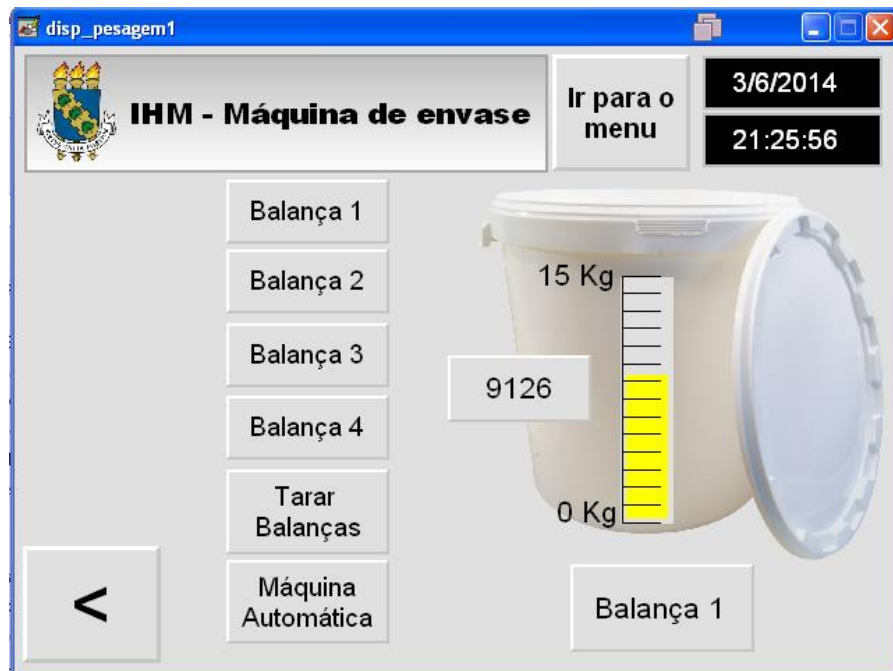
Setpoints Pesagem	15 Kg	3 Kg
Balança 1	14510	2650
Balança 2	13540	2580
Balança 3	13520	2500
Balança 4	13770	2470
Peso de antecipação	450	0

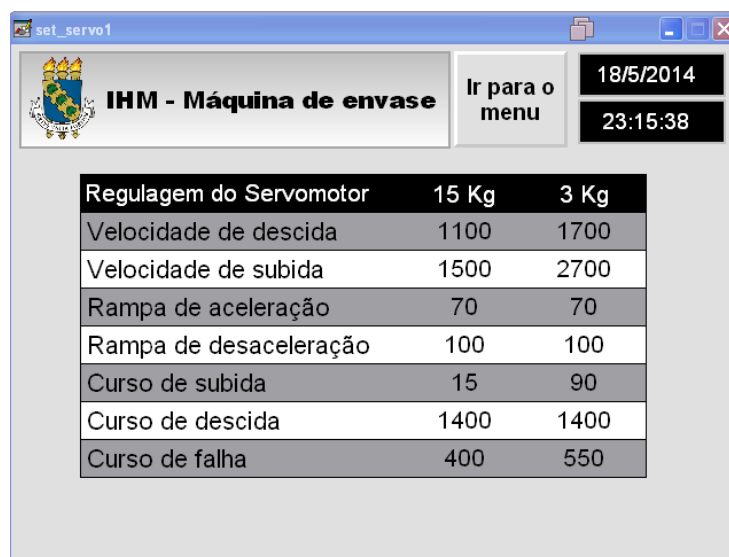
Fonte: Ilustração própria obtida com o auxílio do *software Factorytalk View ME*.

Diferentemente das outras telas de configurações, o *display* de ajustes da pesagem utiliza *tags* internas, pois todos os valores mostrados estão sobre influência de um fator de escala. Os valores armazenados no CLP são dez vezes maiores que os valores exibidos na IHM. Por isso, foram utilizadas *tags* internas. Apesar de exibir o valor do *setpoint* de cada balança com o fator de correção, foi necessário o uso de expressões associadas a cada mostrador numérico para realizar a correção do valor na hora de armazenar o valor do ajuste informado pelo operador. Para isto, utilizou-se a expressão “?*10”, onde o ponto de interrogação representa o valor digitado pelo usuário no teclado da interface.

No canto inferior da tela mostrada na Figura 6.24, existe um botão do tipo “*GoTo Display*” que abre a segunda tela de ajuste dos parâmetros das balanças nomeada de “disp_pesagem1”. Neste *display*, é permitido ao usuário da interface o monitoramento do valor medido em cada balança em tempo real. Além do acompanhamento dos valores medidos, é possível habilitar o modo “teste” da máquina que permite a utilização das balanças com o restante da máquina parada. O modo de teste da máquina, juntamente com o comando de tarar as balanças via IHM, permite que um peso padrão seja colocado sobre uma das balanças e, por meio da interface, o valor medido pelos sensores da máquina correspondente ao peso inserido seja observado, facilitando o ajuste da balança. Esta tela foi desenvolvida com o intuito de que o operador possa ter uma ideia da dimensão dos valores mensurados

pelas balanças e armazenados no CLP. Este conhecimento é necessário para o ajuste dos *setpoints* exibidos na tela “disp_pesagem”. A Figura 6.25 mostra a segunda tela de ajustes dos parâmetros da balança.

Figura 6.25 – Tela “disp_pesagem1” da aplicação desenvolvida.

Fonte: Ilustração própria obtida com o auxílio do *software Factorytalk View ME*.

O canto inferior esquerdo da tela exibida acima, um botão do tipo “*GoTo Display*” permite o retorno à primeira tela de parâmetros das balanças. Na parte central da tela, encontram-se diversos comandos. Os quatro primeiros botões da faixa central (Balança 1 à Balnaça 4) da tela são do tipo “*Macro Button”*, isto é, permitem a execução de uma macro associada após o clique do usuário. Cada um desses quatro botões seleciona uma balança para ser monitorada no lado direito da tela. A escolha da balança é feita por meio de uma *tag* interna e quatro macros. A *tag* “balanca” é do tipo analógica e pode receber um valor de 1 a 4. Para atribuir o valor dessa *tag* foi necessário o uso de quatro macros: “b1”, “b2”, “b3” e “b4”. Os nomes atribuídos as macros são bastante sugestivos, cada macro atribui um valor de 1 a 4 a *tag* “balanca” de acordo com o indicador numérico do nome do arquivo da macro, isto é, quando a macro “b1” é disparada a *tag* “balanca” recebe o valor 1. O disparo das macros é feito pelos quatro botões que contém o nome da respectiva balança. Ainda na parte central da tela, há um comando para tarar as balanças, implementado por meio de um *push button*, que deve ser utilizado quando a máquina está no modo “teste”. A seleção do modo automático ou “teste” pode ser feita por meio do último botão da faixa central da tela. Esse botão é do tipo

retentivo e, quando habilita o modo “teste” da máquina, possibilita a utilização de alguns comandos (neste caso o comando de tarar as balanças) mesmo com a máquina parada.

No lado direito do *display* “disp_pesagem1”, uma imagem foi utilizada juntamente com um gráfico e um mostrador para indicar o valor medido na balança selecionada. A barra gráfica com preenchimento proporcional ao valor medido foi implementada por meio do elemento *Bar Graphic*. Este elemento permite a observação de um valor de forma gráfica. Sua configuração é composta pela atribuição de valores mínimo e máximo, cores de preenchimento e uma *tag* ou expressão para apontar o valor observado. A conexão do elemento foi estabelecida por meio de uma expressão. Cada balança tem o seu valor de leitura armazenado em um registrador do tipo *long*. A expressão lógica utilizada no elemento gráfico analisa o valor da *tag* interna “balanca” e, de acordo com o valor observado, aponta para o endereço do valor medido pela balança escolhida pelo o usuário. O mostrador numérico posicionado a esquerda da imagem do balde, também tem a sua conexão definida por meio da mesma expressão utilizada no gráfico. Logo abaixo da imagem do balde, um painel indica qual balança está sendo monitorada.

6.4.7 Display de ajustes do servomotor

Por fim, a última tela de ajustes de parâmetros utilizadas na IHM foi a “set_servo1”. Nesta tela, todos os parâmetros utilizados para configuração do servomotor estão disponíveis para modificação e acompanhamento. A respeito dos elementos gráficos e recursos da interface utilizados não há nenhuma novidade a ser abordada na tela de ajuste do servomotor, pois a metodologia aplicada é a mesma que outras telas de configurações.

Figura 6.26 – Tela “set_servo1” da aplicação desenvolvida.

Regulagem do Servomotor	15 Kg	3 Kg
Velocidade de descida	1100	1700
Velocidade de subida	1500	2700
Rampa de aceleração	70	70
Rampa de desaceleração	100	100
Curso de subida	15	90
Curso de descida	1400	1400
Curso de falha	400	550

Fonte: Ilustração própria obtida com o auxílio do *software Factorytalk View ME*.

O *hardware* utilizado e a solução de automação aplicada no controle do servomotor já foram abordados detalhadamente nos capítulos anteriores deste trabalho. Como foi apontado na Figura 6.22, o servomotor é responsável pelo deslocamento vertical do braço mecânico da estação de tampas. Para o controle preciso do posicionamento do braço mecânico do dispensador de tampas, devem ser configurados os cursos de descida, subida e falha. Esses parâmetros correspondem a posição inicial do braço giratório da estação de tampas, posição final e posição em caso de falha. Os outros parâmetros ajustáveis via IHM são: controle da rotação do motor correspondente a velocidade de descida e subida do braço mecânico e a rampa de aceleração e desaceleração do servomotor.

6.4.8 Displays de alarmes

As duas últimas telas desenvolvidas para a IHM foram a “alarmes” e a “alarmes_disp”. Essas telas possuem a dimensão de 629x264 pixels, isto é, são menores que as demais telas da IHM. Essa configuração personalizada foi possível devido ao tipo de posicionamento da tela. Diferentemente das demais telas que substituem o *display* anterior ao serem exibidas, as telas de alarme se sobrepõem as demais telas e não podem ser substituídas.

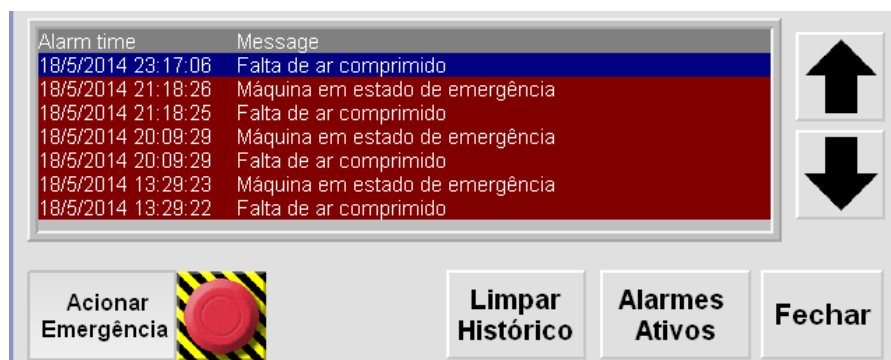
Cada tela de alarme possui comandos voltados para o gerenciamento dos alarmes da máquina. A primeira tela “alarmes” é apresentada na Figura 6.27. O *software Factorytalk View* possui um display padrão para exibição de alarmes. Porém, por questões visuais e de adequação do *display* de alarmes as demais telas da IHM, optou-se por o desenvolvimento de uma tela de alarmes própria. Esta tela está apontada na configuração dos alarmes da IHM e é exibida sempre que um novo alarme é disparado. Na tela “alarmes”, o elemento principal é o *Alarm Banner*. Uma aplicação desenvolvida no *software Factorytalk View*, após a configuração de alarmes, roda um datalog dos alarmes acionados a partir do momento que a interface é executada. O elemento *Alarm Banner* exibe justamente a entrada mais recente do datalog nativo da aplicação, mostrando a data, hora e mensagem personalizada para cada alarme de acordo com as configurações feitas no editor de alarmes do *software*. Além do *banner* de alarmes, os outros elementos encontrados na primeira tela de alertas são: um botão de emergência, um botão de reset, um botão de navegação para a segunda tela de alarmes (Histórico) e um botão de fechamento. O botão de reset está associado a um endereço do CLP que cumpre a função *unlatch* dos alarmes. Esse botão é necessário, pois um alarme, após ser disparado, continua ativo mesmo que a máquina retorne para suas condições normais de operação. O botão histórico exibe a segunda tela de alarmes que possui um relatório mais

detalhado. O último botão desta primeira tela de alertas é do tipo *Close Display*, isto é, ele encerra o display de alarmes e volta para a última tela exibida antes da ocorrência de uma falha.

Figura 6.27 – Tela “alarmes” da aplicação desenvolvida.

Fonte: Ilustração própria obtida com o auxílio do *software Factorytalk View ME*.

A segunda tela de alarmes, só pode ser exibida por meio do comando disponível na tela “alarmes”. A Figura 6.28 mostra os elementos da segunda tela de alarmes. O principal elemento da tela “alarmes_hist” é o objeto *Alarm List*. Diferentemente do *Alarm Banner* que só mostra um alarme por vez, o *Alarm List* mostra uma listagem completa de alarmes de acordo com o datalog de alertas da aplicação. Essa lista pode ser personalizada com diversas colunas de informações, como reconhecimento de alarmes, e animações gráficas de acordo com o estado do alarme. Ao lado do histórico de alarmes, há dois botões para navegação na lista de alertas. Esses botões são configurados para interagir com o objeto em foco que no caso da segunda tela de alertas é o *Alarm List*. Os botões implementados no canto inferior da tela são basicamente os mesmos da primeira tela com a modificação do botão de navegação que retorna para o *display* “alarmes” e do botão “Limpar Histórico”. Esse último botão efetua uma limpeza no datalog de alarmes da aplicação.

Figura 6.28 – Tela “alarmes_hist” da aplicação desenvolvida.

Fonte: Ilustração própria obtida com o auxílio do *software Factorytalk View ME*.

6.5 Conclusões

No último capítulo do desenvolvimento, foram abordados diversos conceitos e recursos do desenvolvimento de interface gráfica. Neste capítulo, foi documentado todas as etapas seguidas durante o desenvolvimento de uma IHM feita para interagir com uma máquina envasadora. A escolha do desenvolvimento de uma interface ao invés de uma aplicação *ladder*, foi baseada na possibilidade de simulação, isto é, a IHM pode ser simulada e testada em uma situação muito próxima da real. Já uma aplicação *ladder* necessita de uma implementação real completa para validar o seu desenvolvimento. Portanto, a simulação de uma interface em conjunto com o estudo de uma aplicação em *ladder* real torna um trabalho introdutório as diversas áreas de automação muito mais completo e proveitoso. A interface desenvolvida, apesar de não ter a complexidade de um sistema SCADA, utiliza os mais diversos recursos oferecidos pelo *software* Factorytalk View, cumprindo o objetivo de mostrar uma visão geral de um projeto de automação que possibilita o desenvolvimento de aplicações mais complexas no futuro. A simulação da interface proposta neste trabalho foi realizada com sucesso permitindo que a IHM interagisse de forma integral com a aplicação do caso em estudo

7 CONCLUSÃO

A abordagem sobre as principais ferramentas computacionais de umas das maiores desenvolvedoras do setor de automação (*Rockwell Automation*) e suas respectivas funções pode ser destacada como primeiro ponto positivo do trabalho, permitindo uma maior familiaridade com as soluções de automação implementadas atualmente no setor industrial. Essa abordagem, em conjunto com os conceitos básicos de *hardware* e linguagem de programação apresentados neste trabalho, permitiram o desenvolvimento de interface que teve suas funcionalidades validadas por meio de simulação. Durante o desenvolvimento deste trabalho, as principais dificuldades enfrentadas foram algumas incompatibilidades dos softwares utilizados com o sistema operacional *Windows 7*, sendo necessária a utilização de uma máquina virtual com sistema mais antigo para desenvolvimento da interface. Além disso, o fato do desenvolvimento da interface ser baseado em uma simulação de uma aplicação real dificulta bastante o entendimento da aplicação, compreensão dos recursos necessários e funcionais a serem implementados na IHM e realização de testes. Os diversos conceitos abordados no decorrer do trabalho servem como base para trabalhos futuros como a implementação real de uma IHM, desenvolvimento de uma aplicação ladder ou apresentação de outros casos que utilizem ferramentas computacionais de outras grandes desenvolvedoras.

Enfim, no decorrer deste trabalho diversas áreas da automação foram abordadas, como hardwares comumente utilizados, conhecimento básico da linguagem de automação mais utilizada, estudo de uma rede industrial e um programa ladder já implementados e desenvolvimento completo de uma interface para interagir com a máquina envasadora estudada. Toda a base teórica abordada na primeira parte do trabalho foi fundamental para o sucesso do desenvolvimento e simulação da interface gráfica proposta. Após a realização deste trabalho, foi adquirido um conhecimento básico sobre automação que, de certa forma, não foi proporcionado pelo curso de graduação em engenharia elétrica enfrentado pelo autor e é fundamental para o sucesso profissional de um engenheiro eletricista, principalmente, no ramo industrial.

REFERÊNCIAS

[1] ROCKWELL AUTOMATION. **Slc 500 programmables controllers**. [S.l.]: [s.n.], 2006.

[2] Petruzella, Frank D. **Programmable Logic Controllers**. 4. ed. Nova Iorque: McGraw-Hill, 2010. 385 p.

[3] ALLEN-BRADLEY. **Micrologix 1200 and micrologix 1500 programmable controllers**: Instruction set refenrece manual. [S.l.]: [s. n.], 2000.

[4] ROCKWELL AUTOMATION. **Rslinx classic:** Getting results guide**.** [S.l.]: [s.n.], 2011.

[5] ROCKWELL AUTOMATION. **RSLogix 500:** Getting results guide**.** [S.l.]: [s.n.], 2010.

[6] ROCKWELL AUTOMATION. **RSEmulate 500:** Getting results guide**.** [S.l.]: [s.n.], 2004.

[7] ROCKWELL AUTOMATION. **RSNetWorx:** Getting results guide**.** [S.l.]: [s.n.], 2012.

[8] HOTTINGER BALDWIN MESSTECHNIK GMBH. **Data sheet**: PW15AH. [S.l.]: [s.n.], [200-?].

[9] HOTTINGER BALDWIN MESSTECHNIK GMBH. **Data sheet**: AED9401A. [S.l.]: [s.n.], [200-?].

[10] ODVA. **Planning and installation manual**: DeviceNet Cable System. [S.l.]: [s.n.], 2002.

[11] MACKAY, Steve et al. **Practical Industrial Data Networks**: Design, Installation and Troubleshooting. Oxford: Elsevier, 2004.

[12] SEW EURODRIVE. **Manual**: DFD11B DeviceNet. [S.l.]: [s.n.], 2007.

[13] HOTTINGER BALDWIN MESSTECHNIK GMBH. **Operating manual**: Digital Transducer Electronics AED9401A. [S.l.]: [s.n.], [200-?].

[14] SEW EURODRIVE. **Manual**: MOVIDRIVE MDX61B "Table Positioning" Application. [S.l.]: [s.n.], 2007.

[15] ROCKWELL AUTOMATION. **Factorytalk View Machine Edition:** User's Guide**.** [S.l.]: [s.n.], 2013.

APÊNDICE A – FLUXOGRAMA DO CÓDIGO DE PESAGEM

Figura A.1 – Fluxograma programa ladder de dosagem

Início - Dosagem

Os Seguintes paramêtros são configurados de acordo com o balde:
- Peso Objetivo
- Tempo de acionamento do motor da esteira de balanças
- Peso de antecipação para abertura da válvula

Balança 1 e 2 com balde

Balança 3 e 4 com balde

Levanta a balança e realiza a tara

Levanta a balança e realiza a tara

Dosagem está perto de ser concluída ?

Sim

Dosadoras 3 e 4 estão em operação?

Dosadoras 1 e 2 estão em operação?

Sim

Dosagem está perto de ser concluída ?

Não

Não

Válvulas dosadoras abertas até o valor meta ser atingido

Válvulas dosadoras abertas até o valor meta ser atingido

Não

Não

Dosagem Concluída ?

Dosagem Concluída ?

Sim

Sim

Liga o motor da esteira 1 e 2

Liga o motor da esteira 3 e 4

Reseta bits virtuais de estado

Fonte: Ilustração própria.

ANEXO A – TABELA COM DESCRIÇÃO DE I/O

Tabela B.1 – Entradas do controlador.

Endereço	Símbolo	Descrição
I:0/0	PRESSOSTATO	Pressostato de ar comprimido
I:0/1	EMERGENCIA	Botão de emergência
I:0/2	LIGA	Botão de liga máquina
I:0/3	DESLIGA	Botão de desliga máquina
I:0/4	TESTE	Chave coloca máquina em teste / automático
I:0/5	RESET	Botão de reset de alarmes
I:0/6	LIGA_DOSADOR	Chave liga / desliga dosador
I:0/7	RESERVA_01	
I:0/8	DISP_BALDES	Liga / desliga dispensador de baldes
I:0/9	DISP_TAMPAS	Liga / desliga dispensador de tampas
I:0/10	ALIM_BALDES_MAN	Alimentador de baldes em manual
I:0/11	ALIM_BALDES_AUT	Alimentador de baldes em automático
I:0/12	ALIM_TAMPAS_MAN	Alimentador de tampas em manual
I:0/13	ALIM_TAMPAS_AUT	Alimentador de tampas em automático
I:0/14	BALDE_3K	Seleciona balde de 3 quilos
I:0/15	BALDE_15K	Seleciona balde de 15 quilos
I:2/0	SNS_PARADOR1	Sensor balde no parador 1
I:2/1	SNS_PARADOR2	Sensor balde no parador 2
I:2/2	SNS_PARADOR3	Sensor balde no parador 3
I:2/3	SNS_PARADOR4	Sensor balde no parador 4
I:2/4	SNS_EMPURRADOR1	Sensor empurrador 1 recuado
I:2/5	SNS_EMPURRADOR2	Sensor empurrador 2 recuado
I:2/6	SNS_EMPURRADOR3	Sensor empurrador 3 recuado
I:2/7	SNS_EMPURRADOR4	Sensor empurrador 4 recuado
I:2/8	SNS_SEP_BALDE_2	Sensor magnético separador de baldes 2 para cima
I:2/9	SNS_BALDE_GR_TAMPA	Sensor estação de tampa balde grande
I:2/10	SNS_BALDE_PEQ_TAMPA	Sensor estação de tampa balde pequeno
I:2/11	SNS_BALDE_PRES	Balde no pressionador de tampas
I:2/12	RESERVA_02	
I:2/13	RESERVA_03	
I:2/14	RESERVA_04	
I:2/15	VACUOSTATO	Vácuostato colocador de tampas

Fonte: Tabela própria.

Tabela B.2 – Saídas do controlador

Endereço	Símbolo	Descrição
O:0/0	EST_ALIMENTACAO	Esteira de alimentação de baldes
O:0/1	EST_BALANCA1	Esteira das balanças 1 e 2
O:0/2	EST_BALANCA3	Esteira das balanças 3 e 4
O:0/3	EST_SAIDA	Esteira de saída de baldes
O:0/4	SEL_VEL_INV	Seleção de velocidade para os inversores
O:0/5	PARADOR1_3K	Parador 1 de balde de 3k e 5k
O:0/6	PARADOR2_3K	Parador 2 de balde de 3k e 5k
O:0/7	PARADOR3_3K	Parador 3 de balde de 3k e 5k
O:0/8	PARADOR4_3K	Parador 4 de balde de 3k e 5k
O:0/9	RESERVA_06	
O:0/10	LAMP_LIGADA	Lâmpada indicadora máquina ligada
O:0/11	LAMP_ALARME	Lâmpada indicadora de alarme
O:0/12		
O:2/3	CORTE_DOSADOR	Válvula de corte do dosador
O:3/0	FREIO_INF_BALDE_15K	Freio inferior baldes de 15k
O:3/1	FREIO_SUP_BALDE_15K	Freio superior baldes de 15k
O:3/2	FREIO_INF_BALDE_3K	Freio inferior baldes de 3 e 5k
O:3/3	FREIO_SUP_BALDE_3K	Freio superior baldes de 3 e 5k
O:3/4	PRENDE_BALDE_15K	Prendedor de balde de 15k (pegador)
O:3/5	DISP_BALDE_15K	Dispensador de balde de 15k (freio)
O:3/6	SEPARA_BALDE_15K	Separador de balde de 15k (elevador)
O:3/7	PUXA_BALDE_15K	Puxador de balde de 15k
O:3/8	PRENDE_BALDE_3K	Prendedor de balde de 3k (pegador)
O:3/9	PARADOR_BALDE3_PRESS	Parador de balde de 3k no pressionador de tampas
O:3/10	DISP_BALDE_3K	Dispensador de balde de 3k (freio)
O:3/11	DISP_BALDE_5K	Dispensador de balde de 5k (freio)
O:3/12	SEPARA_BALDE_3K	Separador de balde de 3 e 5k (elevador)
O:3/13	PUXA_BALDE_5K	Puxador de balde de 5k
O:3/14	VACUO_TAMPA_15K	Vácuo pegador de tampa de 15k
O:3/15	VACUO_TAMPA_3K	Vácuo pegador de tampa de 3k e 5k
O:3/16	FREIO_INF_TAMPA_3K	Freio inferior tampas de 3k
O:3/17	FREIO_SUP_TAMPA_3K	Freio superior tampas de 3k
O:3/18	FREIO_INF_TAMPA_5K	Freio inferior tampas de 5k
O:3/19	FREIO_SUP_TAMPA_5K	Freio superior tampas de 5k
O:3/20	SELEC_VARAO	Seleção do varão para alimentador de tampas 3k e 5k
O:3/21	GIRA_TAMPA_3K	Cilindro gira tampa de 3 e 5k
O:3/22	GIRA_TAMPA_15K	Cilindro gira tampa de 15k
O:3/23	DISP_TAMPA_3K	Dispensador de tampas de 3 e 5k
O:3/24	RESERVA_07	
O:3/25	FREIO_INF_TAMPA_15K	Freio inferior tampas de 15k
O:3/26	FREIO_SUP_TAMPA_15K	Freio superior tampas de 15k

O:3/27	DISP_TAMPA_15K	Dispensador de tampas de 15k
O:3/28	RESERVA_08	
O:3/29	PARADOR_BALDE_3K_TP	Parador de balde de 3 e 5k no dispensador detampas
O:3/30	PARADOR_BALDE_15K_T P	Parador de balde de 15k no dispensador de tampas
O:3/31	PUXA_BALDE_3K	Puxador de balde de 3k
O:4/0	PARADOR_BALDE_PRESS	Parador de baldes 15k no pressionador de tampas
O:4/1	LEV_BALDE_PRESS	Levantador de baldes na estação do pressionador detampas
O:4/2	PRESSIONADOR_TAMPAS	Pressionador de tampas
O:4/3	LEV_BALANCA_1_2	Levantador das balanças 1 e 2
O:4/4	LEV_BALANCA_3_4	Levantador das balanças 3 e 4
O:4/5	VALV_RETORNO	Válvula de retorno geral de produto
O:4/6	VALV_DOS_4	Válvula do dosador 4
O:4/7	VALV_DOS_3	Válvula do dosador 3
O:4/8	VALV_DOS_2	Válvula do dosador 2
O:4/9	VALV_DOS_1	Válvula do dosador 1
O:4/10	RESERVA	
O:4/11	PARADOR2_15K	Parador 2 balde de 15k estação dos empurradores
O:4/12	PARADOR3_15K	Parador 3 balde de 15k estação dos empurradores
O:4/13	PARADOR4_15K	Parador 4 balde de 15k estação dos empurradores
O:4/14	EMPURR_1_DOSADOR	Empurrador 1 e 2 para estação de dosagem
O:4/15	EMPURR_3_DOSADOR	Empurrador 3 e 4 para estação de dosagem
O:4/22		
O:5/0	FREIO_INF2_BALDE_3K	Freio inferior 2 do balde de 3k
O:5/1	FREIO_SUP2_BALDE_3K	Freio superior 2 do balde de 3k
O:5/2	SEPARA_BALDE2_3K	Separador de balde de 3 e 5k (elevador)
O:5/3	PUXA_BALDE2_3K	Puxador de balde de 3k
O:5/4	PRENDE_BALDE2_3K	Prendedor de balde de 3k (pegador)
O:5/5	DISP_BALDE2_3K	Dispensador de balde de 3k (freio)
O:5/6	DISP_BALDE2_5K	Dispensador de balde de 5k (freio)

Fonte: Tabela própria.

ANEXO B – ESQUEMA DE LIGAÇÃO DO SERVODRIVE

Figura C.1 – Ligações do servodrive MDX61B

Fonte: Ilustração própria.